h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbmn
FACHBEREICH MATHEMATIK
UND NATURWISSENSCHAFTEN

Hochschule Darmstadt

Fachbereich Mathematik und Naturwissenschaften & Informatik

Visual Attention in Human Action Recognition

Abschlussarbeit zur Erlangung des akademischen Grades

Master of Science (M. Sc.)

im Studiengang Data Science

vorgelegt von

Georg Frey

Matrikelnummer: 768145

Referent	:	Prof. Dr. Elke Hergenröther
Korreferent	:	Prof. Dr. Andreas Weinmann
Zusätzliche Betreuung	:	M. Sc. Fabian Sturm

Ausgabedatum	:	14.06.2021
Abgabedatum	:	29.11.2021

DECLARATION

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, 29.11.2021

Georg Frey

ABSTRACT

Modern machine learning faces a number of challenges. Artificial neural networks have long been known for their difficult interpretability, which has earned them some notoriety as black-box-models. With ever-increasing amounts of data and growing neural network depths, this issue has only intensified. This is especially apparent in computer vision and with video data. Among large amounts of noisy data, only a tiny fraction of features may eventually contain relevant information.

This makes machine learning applications in computer vision not only costly and resource-intensive to train, but also complicates the analysis of trained networks. Unwanted behavior and its causes may be difficult or infeasible to investigate. This is problematic, as neural networks can develop an unintended focus on secondary features. For example, they could erroneously learn to detect rails, streets or clouds when they were meant to classify trains, cars and planes.

Attention mechanisms present a promising approach to address these problems. Like artificial neural networks themselves, they are inspired by nature. They attempt to mimic how humans perceive their environment. Similar to how humans may pay attention to an object of interest in a picture, attention mechanisms aim to incentivize a neural network to concentrate its calculations during on relevant features.

This thesis investigates how different implementations of visual attention mechanisms can improve results in the task of human action recognition as a subtask of video classification. Furthermore, the advantages of attention for interpreting the behavior of neural networks are researched.

ZUSAMMENFASSUNG

Moderne Machine-Learning-Techniken sind mit einer Vielzahl von Herausforderungen konfrontiert. Künstliche neuronale Netze sind seit langem dafür bekannt, schwer interpretierbar zu sein und werden stellenweise kritisch als Black-Box-Modelle bezeichnet. Dank des stetigen Anstiegs von Datenmengen und dem einhergehenden Wachstum von neuronalen Netzwerkarchitekturen, hat sich dieses Problem zunehmend intensiviert. Besonders offensichtlich ist diese Problematik im Bereich Computer Vision und bei der Arbeit mit Videodatensätzen. Die hier behandelten Daten sind besonders umfangreich und enthalten große Mengen von Noise und nur ein Bruchteil der Daten enthalten relevante Features.

Dies macht das Training von Machine-Learning-Applikationen im Bereich Computer Vision nicht nur Kosten- und Ressourcen-intensiv, es erschwert auch die Analyse der Modelle. Unerwünschtes Verhalten kann nur schwer untersucht und festgestellt werden. Dies ist problematisch, da neuronale Netze einen ungewünschten Fokus auf unbeabsichtigte Sekundär-Features entwickeln können. Beispielsweise könnte sie fälschlicherweise lernen Schienen, Straßen oder Wolken zu erkennen, obwohl sie Züge, Fahrzeuge und Flugzeuge klassifizieren sollten.

Sogenannte Aufmerksamkeit (engl. attention) stellt einen vielversprechenden Lösungsansatz für diese Probleme dar. Wie künstliche neuronale Netze sind diese Mechanismen auch von der Natur inspiriert und versuchen nachzuahmen, wie Menschen ihre Umgebung wahrnehmen. So wie Menschen möglicherweise in einem Bild ihre Aufmerksamkeit auf ein bestimmtes Objekt lenken, so sollen Aufmerksamkeitsmechanismen ein neuronales Netz dazu bewegen, sich während des Trainings auf möglichst relevante Features zu konzentrieren.

Diese Masterarbeit untersucht, wie verschiedene Implementierungen von visuellen Aufmerksamkeitsmechanismen die Ergebnisse bei der Erkennung von menschlichen Handlungen in Videodaten verbessern können. Weiterhin werden die Vorteile von Aufmerksamkeit bei der Interpretation des Verhaltens von neuronalen Netzen untersucht.

The true art of memory is the art of attention

— Samuel Johnson

ACKNOWLEDGMENTS

This thesis was written while working at, and with the support of, ORDIX AG[1]. I thank all my colleagues for their assistance and their patience.

The simple act of paying attention to each other can take you a long way.

1 https://www.ordix.de/

CONTENTS

I THESIS

1 INTRODUCTION 2
- 1.1 Motivation . . . 2
- 1.2 Outline . . . 3
- 1.3 Scope . . . 3

2 FUNDAMENTALS 5
- 2.1 Human Action Recognition . . . 5
- 2.2 Neural Network Basics . . . 6
  - 2.2.1 2-Dimensional Convolutional Neural Networks . . . 7
  - 2.2.2 3-Dimensional Convolutional Neural Networks . . . 9
  - 2.2.3 Residual Neural Networks . . . 9
  - 2.2.4 Long Short-Term Memory Networks . . . 10
- 2.3 Video Classification Architectures . . . 12
  - 2.3.1 2-Dimensional Convolutional Neural Networks . . . 12
  - 2.3.2 3-Dimensional Convolutional Neural Networks . . . 13
  - 2.3.3 Long Short-Term Memory Networks . . . 15
  - 2.3.4 Two-Stream Networks . . . 15
  - 2.3.5 (2+1)D CNN . . . 17
- 2.4 Attention . . . 18
  - 2.4.1 Taxonomy . . . 18
  - 2.4.2 Learn to pay Attention . . . 20
  - 2.4.3 Attention-Gated Network . . . 21
  - 2.4.4 Residual Attention Network . . . 23
  - 2.4.5 Convolutional Block Attention Module . . . 25
- 2.5 Post-Hoc Attention . . . 27
- 2.6 Video Datasets . . . 28
  - 2.6.1 HMDB51 . . . 29
  - 2.6.2 UCF101 . . . 30
  - 2.6.3 Other Video Datasets . . . 30
- 2.7 Image Datasets . . . 31
- 2.8 Data Preprocessing . . . 32
  - 2.8.1 Image Data Augmentation . . . 32
  - 2.8.2 Optical Flow . . . 33
- 2.9 Evaluation Criteria . . . 33

3 RELATED WORK 34
- 3.1 Attention . . . 34
- 3.2 Architectures . . . 35
- 3.3 Approaches . . . 37
- 3.4 Other Research . . . 38

4 CONCEPTION 40
4.1 Approach . . . 40
4.2 Selected Attention Mechanisms . . . 41
4.3 Selected Video Architectures . . . 42
4.4 Backbone Architecture . . . 43
4.5 Implementation and Benchmarks . . . 45
5 IMPLEMENTATION 46
5.1 Hardware Specifications . . . 46
5.2 Software Overview . . . 46
5.3 Project Overview . . . 47
5.4 Backbone Prototypes . . . 47
5.5 Video Classification Models . . . 50
5.6 Attention Visualization . . . 51
6 FINDINGS 53
6.1 Training . . . 53
6.2 Attention . . . 54
6.3 Benchmarks . . . 57
7 DISCUSSION 58
7.1 Attention Modules . . . 58
7.1.1 Learn to pay Attention . . . 58
7.1.2 Attention-Gated Network . . . 59
7.1.3 Residual Attention Network . . . 59
7.1.4 Convolutional Block Attention Module . . . 59
7.2 Challenges . . . 60
7.3 Relation to Post-hoc Attention . . . 60
8 CONCLUSION AND OUTLOOK 61
8.1 Conclusion . . . 61
8.2 Outlook . . . 61

II APPENDIX
Images . . . 63
Bibliography . . . 64

LIST OF FIGURES

Figure 2.1 Simplified illustration of a residual 2D-CNN building block used in residual learning. . . . 9
Figure 2.2 Illustration of a LSTM-unit . . . 11
Figure 2.3 Fusion approaches. Red, green, blue and yellow boxes signify convolutional, normalization, pooling and dense layers respectively. White and gray boxes represent frames and selected frames of the video. Image source: “Large-scale video classification with convolutional neural networks” by Karpathy et al. . . . 12
Figure 2.4 Multiresolution CNN. Red, green, blue and yellow signify convolutional, normalization, pooling and dense layers respectively. Image source: “Large-scale video classification with convolutional neural networks” by Karpathy et al. . . . 14
Figure 2.5 Illustration of the 3D-CNN architecture by Tran et al., leveraging small kernels. The numbers signify the number of filters or dense neurons. Red, blue and yellow signify convolutional, pooling and dense layers respectively. The final layer is a softmax layer. . . . 14
Figure 2.6 Yue-Hei Ng et al. combine 2D-CNN to process spatial information and LSTM to process temporal information. Image source: “Beyond short snippets: Deep networks for video classification” by Yue-Hei Ng et al. . . . 15
Figure 2.7 Two-Stream architecture using a simple backbone architecture. Image source: “Two-stream convolutional networks for action recognition in videos” by Simonyan and Zisserman. . . . 16
Figure 2.8 Illustration of a regular 3D-CNN block (left) and a factorized (2+1)D block (right) . . . 17
Figure 2.9 "Learn to Pay Attention"-modules applied to the VGG16 architecture. Image source: “Learn to pay attention” by Jetley et al. . . . 20
Figure 2.10 Gated attention unit. Here g has the spatial dimensions 3x3 and local features (here F) are subdivided accordingly into a grid. Image source: “Attention-gated networks for improving ultrasound scan plane detection” by Schlemper et al. . . . 23
Figure 2.11 Overview of the residual attention network architecture. Image source: “Residual attention network for image classification” by Wang et al. . . . 24
Figure 2.12 Convolutional Block Attention Module applied to convolutional features F. Image source: “Cbam: Convolutional block attention module” by Woo et al. . . . 26
Figure 6.1 Training curve (top) and test curve (bottom) for the CIFAR-100 dataset. . . . 53

Figure A.1 A CIFAR-100 image of the class *cattle* 63
Figure A.2 A Imagenette image of the class *parachute* 63

LIST OF TABLES

Table 2.1 Results of various convolution filters. 8
Table 2.2 Taxonomy of attention after Chaudhari et al.[7] 19
Table 4.1 Selected attention mechanisms. 42
Table 4.2 Comparison of backbone architectures when applied to the ImageNet dataset. The assessment of the complexity may be subjective. 44
Table 5.1 Comparison of ResNetV2 backbone architecture implementations for 100 classes 49
Table 5.2 Comparison of MobileNet backbone architecture implementations for 100 classes 50
Table 6.1 Comparison of test scores of the backbone prototypes. 54
Table 6.2 Attention maps extracted from the models for the CIFAR-100 dataset. 55
Table 6.3 Attention maps extracted from the models for the Imagenette dataset. 56

GLOSSARY

3D-CNN	2-Dimensional Convolutional Block Attention Module
3D-CNN	3-Dimensional Convolutional Block Attention Module
API	Application Programming Interface
BERT	Bidirectional Encoder Presentations from Transformers
CAM	Class Activation Mapping
CBAM	Convolutional Block Attention Module
CNN	Convolutional Neural Network
FLOPs	Floating-Point Operations
GAN	Generative Adversarial Network
Grad-CAM	Gradient-weighted Class Activation Mapping
L2PA	Learn to pay Attention
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
NLP	Natural Language Processing
RAM	Random-Access Memory
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SAGAN	Self-Attention Generative Adversarial Network
STIP	Space-Time Interest Points
I3D	Two-Stream Inflated 3D-ConvNet

Part I

THESIS

1 INTRODUCTION

The roots of artificial neural networks date back at least to the 1950s.[45] But thanks to increasingly cheap hardware and the emergence of more and more efficient use of graphics processing units, they have become a crucial technology for a number of research fields.[9] At the present day, most state of the art image-recognition software is based or relies heavily on convolutional neural networks.[43] Yet, many of their fundamental drawbacks are still present. They require sizable amounts of data and computational power to be trained. Furthermore, their large number of trainable weights make them notoriously difficult to analyze.[51]

1.1 MOTIVATION

A promising concept in dealing with the problems of artificial neural networks is the concept of attention. Like a human would focus his or her attention on a specific object in an image or a word in a sentence, attention mechanisms aim to force a neural network to focus its calculations on a subset of relevant features. This may address all the issues mentioned above while also potentially improving the model quality.[7]

Although this idea is not necessarily a new one and can be traced back to the 1960s [42][68][7], practical implementations have recently gained in popularity, thanks in part to the notable work of Bahdanau, Cho, and Bengio[2] and Vaswani et al.[63]. Even though their work is primarily on natural language processing, the same principles and underlying techniques can be applied to a variety of tasks, including image classification [29] and video classification[52].

Importantly, video classification and its subfield of human action recognition suffer especially from the general problems of artificial neural networks. Video data are relatively large, and computer vision models in turn also have a high number of trainable weights. If attention mechanisms can improve the efficiency of video classification models, it stands to reason they profit proportionally more than smaller models with fewer parameters would. Likewise, numerous parameters may also make a model more difficult to analyze. For the same reasoning, it would be beneficial if video models could benefit from additional analytic tools like attention.

Furthermore, processing video data in general is a topic of ongoing research. Improving the effectiveness and quality of various video models would be valuable, if it can be accomplished with attention mechanisms.

1.2 OUTLINE

This thesis will begin by describing the general task of human action recognition as a subset of automated video classification. Typical approaches for these tasks will be analyzed.

Modern deep architectures for video classification rely primarily on artificial neural networks. Fundamentally, these architectures can be categorized into 3D-CNN, 3D-CNN and LSTM. Furthermore, a number of more advanced general architectures types exist which build upon these basics.

After discussing the task and identifying appropriate network architectures, the main subject of this thesis, attention, will be formally introduced. To begin with, the broad term is defined and a taxonomy of attention categories is presented. As more types of attention mechanism exist than can be covered in this thesis, the main focus lies on a number of variants which are promising for the task of video classification.

Finally, as with all data-driven applications and neural networks, datasets also play an important role. For this thesis, the HMDB51 [39] and UCF-101 [56] are of particular interest. Both are well-researched video datasets dealing specifically with the classification of human interactions with objects. At the same time, both datasets are of reasonable size, allowing to produce results even with relatively small computational resources.

At this point, the reader will have a thorough understanding of the fundamentals of video classification and attention mechanisms. The first goal of this thesis is to show how various attention mechanisms function and how they can be applied to the problem of video classification. Subsequently, these approaches are to be empirically tested to verify their claimed advantages. Finally, the aim is also to show how attention can help analyze the results and problems a specific model may exhibit during and after training

For the empirical tests, the first goal is to showcase a number of neural network implementations, initially without attention. They will subsequently be used as the basis for the creation of attention versions of these networks. The aim is to keep each two respective networks as similar as possible, with the only difference being the addition or lack of attention-units. Using these networks and the described datasets, a number of benchmarks will be created. Comparing the performance of the neural networks should show measurable improvements when attention is used. It is assumed that different neural network architectures may profit from different implementations of attention mechanisms in different ways. Some architectures and attention mechanisms may also be incompatible with one another. The dataset may also influence the results. Different architectures, attention mechanisms and datasets are used to offset this.

1.3 SCOPE

Due to the broadness of the fields involved and due to the limited availability of computational resources, this thesis will not be able to be exhaustive.

Regarding the types of attention, this work will lie its main focus on visual attention mechanisms. While many types of attention exist and may be relevant for human action recognition, visual attention is especially interesting because of its specificity to computer vision.

Analogously, too many architectures for image classification and video classification exist to be fully investigated and implemented. Therefor the major archetype archetypes will be researched and a number of suitable architectures will be chosen for implementation.

Regarding datasets, the same restrictions apply. Numerous image and video datasets exist, only some of which may be used to evaluate the implementations and produce benchmarks. To optimally use the limited resources, this work will focus on a small number of well-researched, but relatively small datasets.

2

FUNDAMENTALS

This chapter aims to introduce the reader to the fundamentals necessary to understand this thesis. It begins by describing the task and its properties and significance. Subsequently, existing approaches are discussed. As there are numerous interdependent strategies, the order has been chosen to give the reader bottom-up introduction into the various relevant architectures. Afterwards, the central topic of attention is introduced, and several approaches are investigated.

Finally, a few secondary topics are discussed. First, crucially, when working with data-driven algorithms the datasets used must also be carefully examined. Secondly, data processing steps are common when dealing with video data and therefor need to be addressed. Lastly, the evaluation criteria used to benchmark the findings are summarized.

2.1 HUMAN ACTION RECOGNITION

Human action recognition is the task of identifying fully executed human actions in video data and is a considered a fundamental task in computer vision.[35]

It can be regarded as a subset of the broader task of general video classification. More specifically, action recognition can be further divided into action representation and action classification. The former subject attempts to turn pixel data into an effective machine-readable representation. The later task is directly occupied with final attribution of an action.

Furthermore, there are numerous related tasks:[35]

- Action prediction is the classification of task from incompletely executed actions.
- Human-object interaction can be considered a subtask of human action recognition.
- Image and video captioning aim to produce descriptions instead of a class score prediction.
- Image and video segmentation intend to find relevant objects in the data.

Action recognition specifically is of importance, as there are a many applications for this task[35]. For instance, it can be important for surveillance tasks or for quality assurance in production environments. In both situations, a certain subset of action may be disallowed to ensure safety.

Similarly, for autonomous driving, it may be desirable to detect or predict the actions of passersby.

Lastly, for the entertainment industry, gadgets like the Kinect[81] use action recognition for gaming applications.

Lastly, video retrieval applications like search engines may need to classify or rank videos in response to a query.

Video data in general and action recognition specifically face a number of challenges inherent to the data and task.[35] Humans typically have different postures, gaits, quirks or movement behaviors. Different people may perform the same action with an entirely different sequence of movements. This variation is also contained in the datasets and leads to a high variance within each class.

In addition, video data also contains large amounts of noise by default[35]. Video data may be affected by different camera angles, by camera motion, by lighting conditions and by various filming locations. Furthermore, sufficient and reliable annotated data may be mussing in general, as labeling video datasets is costly. Large datasets have often been not annotated by hand, but with automated methods, and contain noisy labels. Lastly, video data contains large amounts of redundant data and is therefor difficult to process. At the same time, only a few key frames may contain information crucial to classify the clip. In summary, video data is often times noisy, has high in variance and has features of highly uneven discriminative quality.

To address these issues, generally two different approaches can be distinguished.

Historically, a number of so-called shallow approaches have existed. They typically focused on finding efficient action representation which can then be classified. Many of those methods relied on hand-crafted features and were thus time-consuming to craft and prone to noise[35] and do not generalize well[30].

This thesis focuses on modern deep architectures which can automatically deduce features and learn representations. Deepness in this context generally suggests artificial neural networks with a high number of trainable variables.

2.2 NEURAL NETWORK BASICS

When dealing with video classification, there are a number of artificial neural network architectures one needs to understand. Most importantly, 2-dimensional convolutional neural networks (2D-CNN) are the fundamental building block when working with image data[35], and therefore they will be discussed first in this chapter.

Subsequently, when approaching other architectures an important distinction which needs to be made is, that videos data can be regarded as a combination of spatial and temporal features. The spatial data describes the image data contained in the individual frames of a video and can be processed with 2-dimensional convolutions. The temporal information is the data arising from the changes between frames and is sequential in nature. This information cannot readily be captured by 2D-CNNs.[35]

Aside from architectural differences, the following networks often also represent different approaches when dealing with these parts of the information contained within the videos.

2.2.1 *2-Dimensional Convolutional Neural Networks*

As the name suggest, convolutional neural networks use convolutions to process image data. Convolutions themselves describe a technique originating in image processing and are also known as kernels, convolution filters, convolution matrix or simply as masks.[74, p. 49]

Although the dimensions may differ, the filter itself typically is a 3x3 matrix. This filter is also called kernel. When applying this kernel to an image, the following steps are performed:[74, pp. 6-7 49]

1. From the original image chose a subsection of pixels ("window") of the same size as the kernel (e.g. a 3x3 matrix).
2. Perform an element-wise multiplication of the selected section and the kernel.
3. Calculate the sum of the elements of the resulting matrix. The value of this summation is the value of the pixel in the resulting image.
4. Repeat steps 1.-3. for all windows of the original image. On each iteration, the relative position of the calculated pixel is maintained in respect to the location of its source window.

Steps 1-3 of this algorithm are referred to as a convolution.[74, pp. 6-7] For an image with multiple channels, like an RGB-image with three color channels, this process can simply be conducted individually for each channel. Also, the attentive reader might notice, that the resulting image will be of a smaller size as the original image, as the kernel cannot be applied to windows at the edges of the image. To circumvent this problem, a number of approaches exist, one of which is zero-padding. In this case, additional pixels with a value of zero are presumed to be around the edges of the image.[74, p. 52] Thus, when using zero-padding, the resulting image can be of the same dimensions as the source.

At first glance, it may be difficult to realize the potential of this technique, but it is extremely potent and versatile. Depending on the kernel, a vast number of image manipulation applications like smoothing[74, pp. 54-61], edge detection[74, pp. 61-68] or sharpening[74, pp. 89-90] can be performed with convolution filters. Table 2.1 displays a few common examples.

Convolutional neural networks use convolution filters in their layers. Most importantly, here the values of the kernels are not fixed values, but variable weights which are learned during training. Furthermore, if the input data contains multiple channels, each channel will be assigned an individual kernel with separate variables.[70] So for a gray-scaled image with a single channel, the number of weights is simply the number of elements: the product of the height and width of the kernel. For an RGB-image with three channels, there are effusively three kernels and the value is tripled. The dimensions of each kernel is also known as kernel size.[43]

As understood from above, a convolution filter may help a neural network to abstract a portion of the information which is contained in the input data. Yet the previous example has also shown how the purpose of each convolution filter is fairly specific.

Description	Filter matrix / Kernel	Resulting image
Original image / identity filter	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Smoothing	$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$	
Sharpening	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Edge Detection	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	

Table 2.1: Results of various convolution filters.

To enable an artificial neural network to learn different representations, a single filter per layer is not enough. To increase the abstraction ability, each convolution layer will typically have a number of individual filters which are applied to the input data.[70] This number is can be referred to as the filter count or simply depth.[43]

Further convolution layer parameters are the application of zero-padding and stride. Here, stride describes the step size between individual application of the convolution kernel. The default convolution calculations can be viewed as having a stride of one. A stride of two means every second window is skipped. A stride of three means the convolution is only applied every three spatial positions, and so on.[70]

The final piece to understanding convolutional neural networks are pooling layers. Fundamentally, they function the same way convolution layers do, but have a few important distinctions and are used for different purposes. Unlike convolution layers, they do not have weights but perform a simple mathematical maximum or averaging operation on their spatial window. Therefore, they are referred to as max-pooling and average-pooling layers, respectively. Additionally, for pooling layers the kernel-size and strides are chosen, so the subregions do not overlap.[70] This can for example be accomplished with a 2x2 kernel and a stride of two.[43] As a consequence, pooling will reduce the size of the spatial dimension of the tensor. This way, pooling helps to reduce the complexity and number of parameters of a neural network.

2.2.2 *3-Dimensional Convolutional Neural Networks*

Fundamentally, the concept of 3-dimensional convolutions is not difficult to grasp if one understands 2-dimensional convolutions. Instead of a matrix with two dimensions (or second order tensor) the kernel simply becomes a third order tensor with three dimensions. Geometrically, one can visualize such a tensor as a cube of values. Instead of a single image, this kernel can now be applied to a sequence of frames. A 3-dimensional operation can thus process temporal information in addition to spatial information.[31] Ji et al. show in 2012 that building deep neural networks based on 3-dimensional convolutions is feasible and yielded competitive results at the time. Yet they relied on handcrafted convolutions in the first layer and deployed a comparatively simplistic architecture by modern standards.

2.2.3 *Residual Neural Networks*

Residual neural networks were originally published in 2016 by He et al. after winning the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in 2015 and are conceptually an important addition to many neural network architectures.[24]

Residual learning as a technique in itself is not necessarily specific to image-processing. Yet, especially image networks have been shown to benefit greatly from this. Especially in comparison to the earlier VGG architectures[55], ResNets were both more effective and efficient, yielding better results at a fraction of the computational cost.

Blocks in residual networks possess an addition shortcut connection between their input and output. The shortcut is typically the identity mapping of the input and is combined with the block output by simple addition. See figure 2.1.

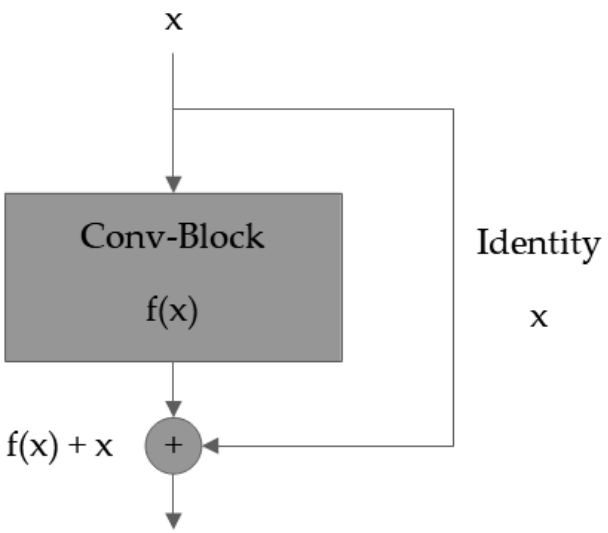


Figure 2.1: Simplified illustration of a residual 2D-CNN building block used in residual learning.

In their reasoning, He et al. cite the degradation problem as the core motivation for residual learning. As conventional network sizes increase, the quality of the model (e.g. accuracy-score) only follows up to a certain saturation point. Following this point, the

accuracy will start to degrade when more layers are added. This is counter-intuitive, as a larger network should theoretically always perform at least as good as a smaller one. Such a network could easily be constructed by a human by copying the parameters of the smaller network and using neutral weights (i.e. identity mapping) in the additional layers. Research has also indicates this problem does not stem from overfitting as one may initially assume.[23][55] Thus, He et al. believed the degradation problem to be rooted in the optimization-difficulty of a network.

In their analysis, the authors verify their residual learning approach on an aggressively large residual network with 1202 layers. They find the shortcuts adequately address the optimization difficulty problem, as the network exhibits only a relatively small difference in training and test error. They argue the remaining difference is due to overfitting.

Concerning the implementation, the authors offer a few options. First, regarding the identity-shortcuts shown in figure 2.1 also consider a linear projection as alternative. Here, the shortcut is additionally parameterized with a trainable weight-matrix. Although the authors deem these parameters to be unnecessary and thus uneconomical in their experimental setup.

Secondly, when the number of convolution filter change, as they typically do within a CNN, the shortcut can not simply be applied to the layer output due to the mismatch in dimensions. Here, two options are offered.

- The simplest approach is simple zero-padding: the missing dimensions are simply filled with zeros.
- As previously discussed, projections can also be used to adjust the shortcut dimensions. For instance, a convolutional layer with a 1x1 and appropriate filter size can be used to adjust the number of channels.

Further research by Veit, Wilber, and Belongie has shown residual blocks help neural networks combat the vanishing gradient problem by decreasing the average length of the path the signal takes through the network.[64] Consequently, even very deep residual networks may exhibit relatively shallow paths. The authors compare this behavior to an ensemble of shallow networks and conclude network depth still to be an open research topic.

2.2.4 *Long Short-Term Memory Networks*

Long short-term memory (LSTM) was proposed by Hochreiter and Schmidhuber in 1997 as an extension to conventional recurrent neural networks RNN.[25] Recurrent neural networks are an important class of artificial neural networks which allow processing sequences of variable length. Thus, they are also of interest when analyzing video data with its sequential temporal information. Long short-term memory networks were primarily invented to address the vanishing and exploding gradient problems common with recurrent network architectures. Hochreiter and Schmidhuber identified the underlying issue as an error back-flow problem occurring during the back-propagation

through time algorithm. They solve this issue by enforcing a constant error flow through the internal states of the LSTM-Units.
An important addition present in modern implementations of LSTM is the forget-gate, proposed by Gers, Schmidhuber, and Cummins.[18] It allows each unit to learn which information of previous cells to discard. In this chapter, solely the extended long short-term memory-units with forget gates are discussed. Figure 2.2 shows the structure of such a LSTM-Unit.

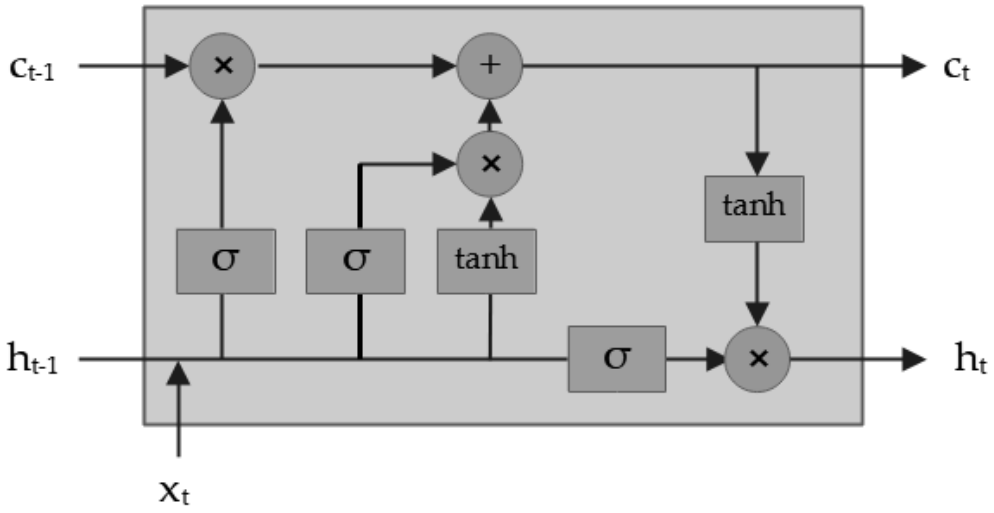


Figure 2.2: Illustration of a LSTM-unit

Each LSTM-unit can have three inputs: the sequential data stored in $\boldsymbol{x_t}$, the cell state of the previous cell $\boldsymbol{c_{t-1}}$ and the hidden state of the previous cell $\boldsymbol{h_{t-1}}$. Here $\boldsymbol{t}$ denotes the time step or position of the input sequence and LSTM-unit. A long short-term memory cell consists of four internal gates, which can most concisely be expressed by equation 2.1 [77].

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ g_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} M \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \tag{2.1}$$

Here $\boldsymbol{i_t}$ is the input gate, $\boldsymbol{f_t}$ is the forget-gate, $\boldsymbol{o_t}$ is the output gate and $\boldsymbol{g_t}$ is the input modulation gate. $\boldsymbol{M}$ denotes a matrix of trainable parameters of appropriate dimensions.

Element-wise multiplication of the gate information is used to calculate the cell state $\boldsymbol{c_t}$ and the hidden state $\boldsymbol{h_t}$ of the current unit. See equations 2.2 and 2.3.[77]

$$c_t = (f_t * c_{t-1}) + (i_t * g_t) \tag{2.2}$$

$$h_t = o_t * tanh(c_t) \tag{2.3}$$

To understand LSTM, one can think of the cell state c_t as the memorized information of each cell and the hidden state h_t as the prediction of each cell. Each cell therefor considers the input sequence x_t, as well as the prediction and memory of the previous cell, for its own predictions and state. Importantly for this thesis, it is also noteworthy that the forget-gate and the input modulation gate fulfil similar roles as attention. The forget-gate functions similar to the noise reduction properties of an attention mask, while the input modulation gate respectively highlights features of interest.

2.3 VIDEO CLASSIFICATION ARCHITECTURES

Now that the fundamental building blocks of neural networks have been explained and a number of basic image classification architectures have been addressed, video classification architectures can be examined.

2.3.1 *2-Dimensional Convolutional Neural Networks*

Despite being designed for image processing, 2-dimensional convolutional neural networks can be adapted for video processing with relatively little effort. Naively, one may train a single network on a selection of frames and combine the predictions by averaging to receive the final classification of a clip. It is trivial to see this approach is fairly limited, as it ignores any temporal context.

Even more simply, Karpathy et al.[33] use a 2D-CNN trained on a singular frame of the video as a baseline for their research into more advanced architectures. Firstly they propose a number of simple architectures they differentiate by the stage the temporal information is fused within the network structure. Appropriately, they name these approaches early, late and slow fusion. See Figure 2.3.

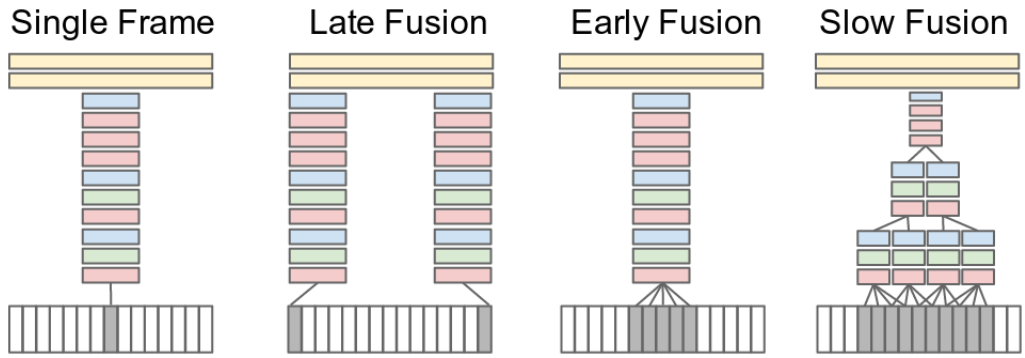


Figure 2.3: Fusion approaches. Red, green, blue and yellow boxes signify convolutional, normalization, pooling and dense layers respectively. White and gray boxes represent frames and selected frames of the video. Image source: "Large-scale video classification with convolutional neural networks" by Karpathy et al.

For early fusion, they use a single 3-dimensional convolution layer (see section **??**) to process a small amount of T=10 frames together. Afterwards, they stack 2D-CNN layers as usual.

The late fusion network classifies individual frames in a distance of 15 frames. Both network paths share the same parameters. The network is finalized by a stack of dense layers which combine the outputs and may analyze temporal information resulting from motion between the frames.

Finally, slow fusion aims to combine the other two approaches and makes heavy use of 3-dimensional convolution layers. Notably, the parallel sections of layers share the weights which each other.

As an alternative approach Yue-Hei Ng et al. experimented with another five 2D-CNN architectures to classify video data.[75] Similarly, they first prepare convolutional features from each video frame by using a regular 2D-CNN and then employ various strategies to combine the network outputs. However, unlike Karpathy et al. they refer do not refer to the combination approaches as fusion, but as pooling.

- **Conv Pooling**. Max-pooling is performed across the CNN-outputs. Afterwards, fully-connected layers are used.
- **Late Pooling**. First fully-connected layers are used to process each CNN-output individually before max-pooling is performed.
- **Slow Pooling**. Max-pooling and fully-connected layers are used in alternation, while forming a hierarchical structure.
- **Local Pooling**. Several distinct max-pooling operations and fully-connected layers are used over a window of frames.
- **Time-Domain Convolution**. A 1D-Convolutional layer is used to combine the 2D-CNN outputs. Afterwards, a max-pooling and fully-connected layers is used.

Regardless of the pooling method, each network is finalized with a softmax layer.

As Karpathy et al. show, it seems only natural to eventually progress from 2-dimensional convolutions to 3-dimensional convolutions to capture the temporal information contained in video datasets.

2.3.2 *3-Dimensional Convolutional Neural Networks*

One such 3D-CNN architecture are the multiresolution CNNs also proposed by Karpathy et al.[33] To use the limited computation power available, they divide the video data into two different streams. First a context stream, which takes a 50% down-sampled version of the input data and secondly the fovea stream, which takes the center-cropped region with 50% of the size of the original input data. With this approach, they aim to make use of the implicit camera-bias, suggesting the object of interest typically to be in the center of the frame. The context-stream on the other hand is intended to efficiently process larger scale context information.

As both stream inputs have the same spatial dimensions, the same network architectures can be used. Finally, both streams are concatenated by two dense layers. See figure

2.4.

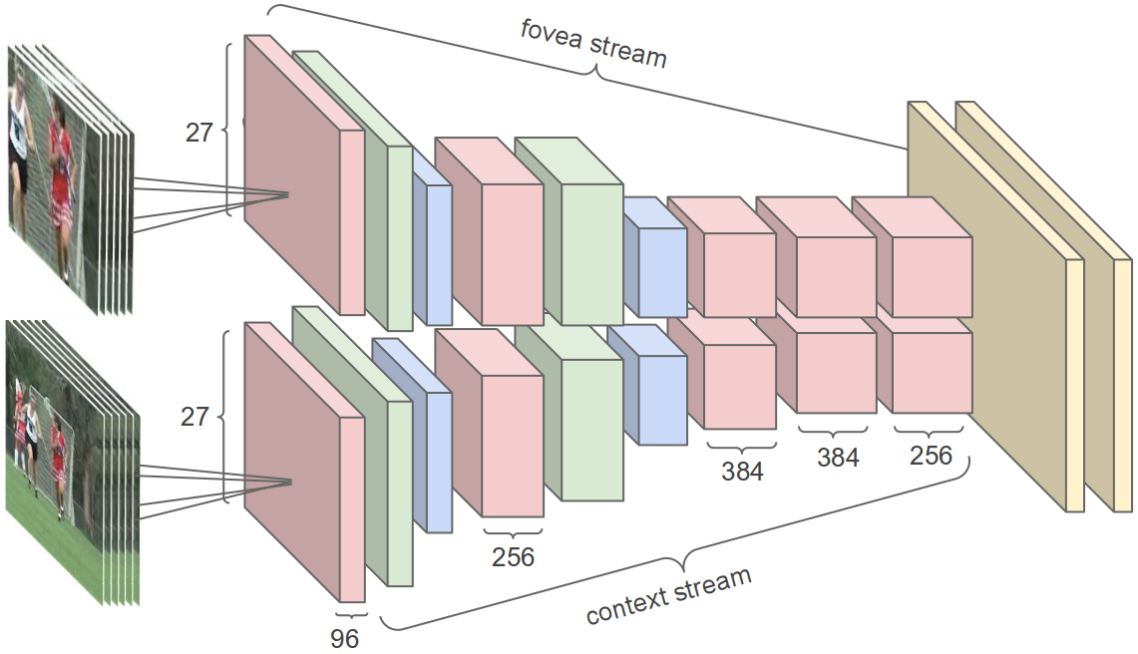


Figure 2.4: Multiresolution CNN. Red, green, blue and yellow signify convolutional, normalization, pooling and dense layers respectively. Image source: "Large-scale video classification with convolutional neural networks" by Karpathy et al.

As an alternative, Tran et al. proposed another 3D-CNN architecture in 2015 which primarily leverages small 3x3x3-kernels.[61] They use a stride of one in all directions and 2x2x2 3-dimensional pooling, which the notable exception to pool 1, which uses 1x2x2 and stride 1x2x2. This way, they intend to preserve the temporal information as long as possible. See figure 2.7.

When trained on the Sports-1M dataset, they used 16 frames of 112×112 resolution and managed to outperform the results of the earlier multiresolution CNN.

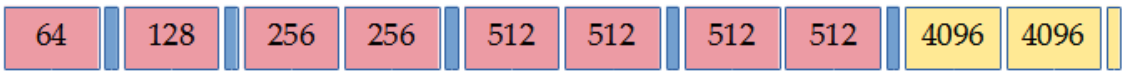


Figure 2.5: Illustration of the 3D-CNN architecture by Tran et al., leveraging small kernels. The numbers signify the number of filters or dense neurons. Red, blue and yellow signify convolutional, pooling and dense layers respectively. The final layer is a softmax layer.

However, when examining 3-dimensional convolutional neural networks, an important observation has to be made. Due to the curse of dimensionality, the number of parameters and computing operations increase exponentially when using cubic convolutions.[21] Efficient use of the available resources therefor becomes a pressing concern.

2.3.3 *Long Short-Term Memory Networks*

Yue-Hei Ng et al. demonstrate how LSTM can relatively easily be used in conjunction with a 2D-CNN to classify human actions.[75] They use a 2D-CNN as a feature extractor, or embedding model, to analyze the spatial information in each frame and use the network activations as a feature vectors. These vectors are fed into a deep, five-layer LSTM with 512 units. See figure 2.6.

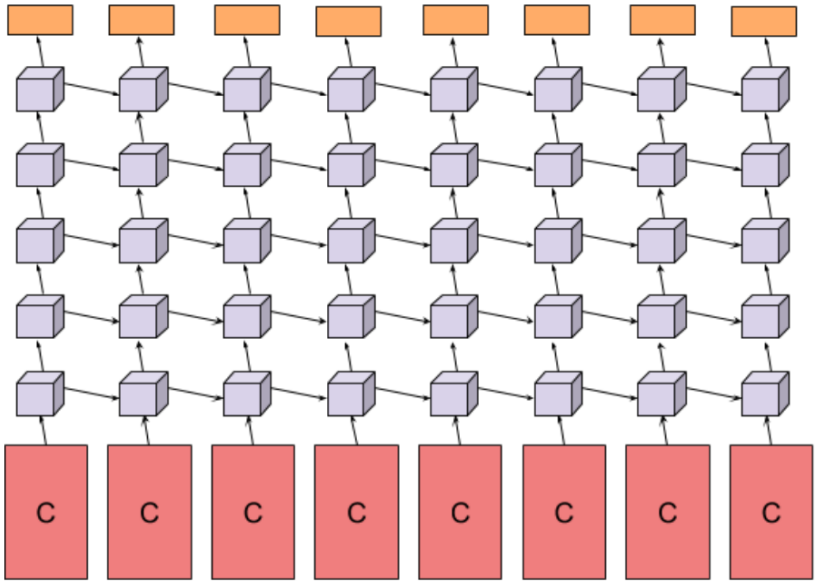


Figure 2.6: Yue-Hei Ng et al. combine 2D-CNN to process spatial information and LSTM to process temporal information. Image source: "Beyond short snippets: Deep networks for video classification" by Yue-Hei Ng et al.

Interestingly, this architecture allows using different image classification architectures as backbones models. Yue-Hei Ng et al. use the AlexNet[38] and GoogLeNet[59] architectures and verify their results on the Sports-1M[33] and UCF101[56] datasets.

2.3.4 *Two-Stream Networks*

Two-Stream architectures were first proposed by Simonyan and Zisserman[54] and are particularly interesting because they allow the usage of conventional 2D-CNN architectures to process temporal information. This has two major benefits:

- As a 2D-CNN used as backbone, advancements in image-classification can directly be used to the benefit of this architecture.
- This architecture is very efficient, as only roughly twice the number of parameters of the backbone architecture are used.

As the name suggest and similarly to the Multiresolution CNN,[33] this architecture addressed video data with two different networks and a late fusion approach. The first spatial network classifies individual frames as earlier described in section 2.2.1. The second

temporal network classifies the optical flow (see section 2.8.2) of the video dataset. Typically, both networks use the same 2D-CNN backbone architecture or versions thereof.

While this approach is fairly straight-forward, two implementation details should be regard. First, Simonyan and Zisserman propose a number of different approaches to utilize and stack the optical flow data. Secondly, there are multiple fusion methods for the individual networks.

Regarding the optical flow, there are a high number of possibilities. Firstly, as a single optical flow frame may yield too little information, the authors suggest stacking the optical flow of multiple consecutive frames. These resulting displacement vectors can easily be treated as individual channels for the usage in a 2D-CNN. This leads to a number of $\mathbf{2}\boldsymbol{L}$ channels, where $\boldsymbol{L}$ is the number of stacked frames.

As a consequence of optical flow stacking, [33] differentiate between regular optical flow stacking and trajectory stacking. The former is the calculation of optical flow between frames 1 and 2, 1 and 3, 1 and 4 ... and so on. Trajectory stacking track a pixel across multiple frames, e.g. from frame 1 to 2, then from 2 to 3 and so on.

Furthermore, there is the option of bidirectional optical flow. As the name suggests here, additional information is provided by additionally calculating the optical flow stack in the opposite direction. This addition yields another two channels of information for each optical flow stack.

To combine the predictions of the two streams, the authors recognize three different fusion methods. Firstly, they address fusion by using dense layers but dismiss this as it was not feasible due to overfitting. Subsequently, they identify fusion by averaging and fusion by using a support vector machine as viable options.

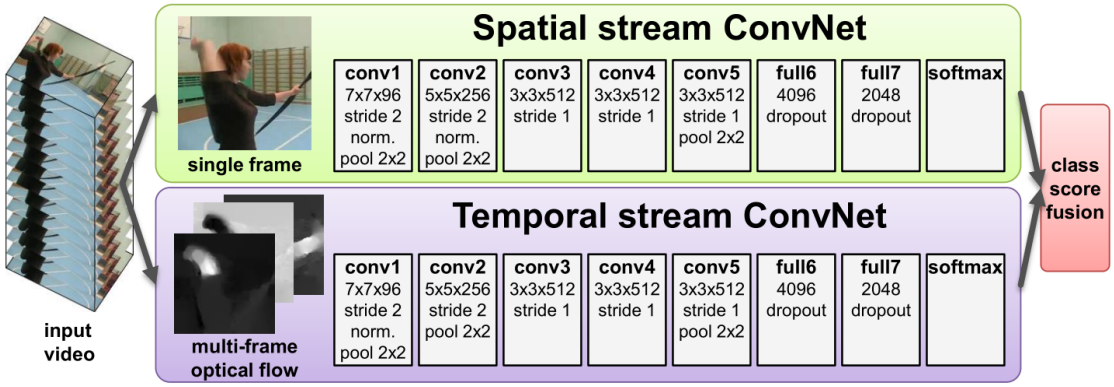


Figure 2.7: Two-Stream architecture using a simple backbone architecture. Image source: "Two-stream convolutional networks for action recognition in videos" by Simonyan and Zisserman.

Simonyan and Zisserman verified their architecture on the HMDB51[39] dataset (see chapter 2.6.1) and UCF101[56] dataset (see chapter 2.6.2).

As can be seen in figure 2.7 the authors use a rather simple backbone architecture for the spatial and temporal network. Importantly, this backbone can be exchanged with more advanced architectures like ResNets[24] (see chapter 2.2.3).

2.3.5 *(2+1)D CNN*

As described in section 2.2.2, 3-dimensional convolutional neural networks suffer from the curse of dimensionality. To combat this problem, Tran et al. in 2018 proposed "(2+1)D" convolutions.[62] At its core, this (2+1)D convolution block simply factorized the 3-dimensional CNN layer into a 2-dimenionsal CNN layer and a subsequent 1-dimensional temporal CNN layer. See figure 2.8.

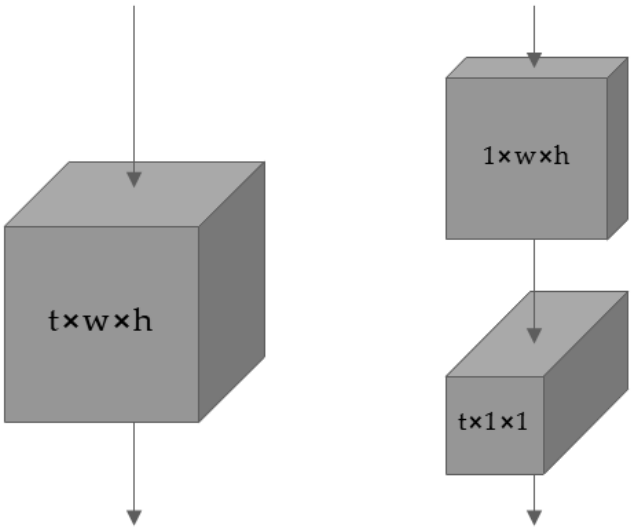


Figure 2.8: Illustration of a regular 3D-CNN block (left) and a factorized (2+1)D block (right)

Despite the apparent simplicity of this change, Tran et al. identify a number of benefits. First, due to this change, the number of rectifier activations effectively doubles. The increased number of nonlinearities allows the network to learn more complex functions. Secondly, when directly compared to 3-dimensional convolutions (2+1)D-convolutions are much easier to optimize, accelerating the training process. Lastly, when the same number of filters are chosen, the (2+1)D-convolutions will have fewer parameters than their 3-dimensional counterparts, making the network more lightweight.

In their experiments, Tran et al. adjust the number of filters in their (2+1)D blocks to be roughly equal to the equivalent 3D block. Thus, they created networks which were easily comparable thanks to their similar number of parameters. Furthermore, they combined their research with the residual neural network concept described in section 2.2.3, creating residual (2+1)D, or R(2+1)D-blocks.

For verification, the authors adapted the ResNet-34 architecture as 34-layer R(2+1)D net. This architecture consistently performed better than their conventional counterparts, both during training and validation. They used the HMDB51[39], UCF101[56], Sports-1M[33] and Kinetics[5] dataset to verify their results and produced state-of-the-art results at time of publication for all four. Furthermore, they also showed various other architectures, like the two-stream architecture (see section 2.3.4) profits from (2+1)D convolutions.

2.4 ATTENTION

In the most general form, attention is a function of keys $\boldsymbol{K}$, a query $\boldsymbol{q}$ and values $\boldsymbol{V}$. The goal is to weight the values in response to how important the key is in relation to the respective query. Mathematically, the first step can be expressed by equation 2.4.[7]

$$c = a(K, q) \tag{2.4}$$

Here, **c** can be understood as compatibility score of the key and query. The function **a** can for instance be the simple dot-product or a function with learnable weights. The attention is directly calculated from this score. See equation 2.5.[7]

$$\alpha = p(a(K, q)) \tag{2.5}$$

The function **p** can for instance be the softmax function to make sure the attention value is within the bounds of **[0, 1]**. When this attention is applied to all values **i**, it can be expressed like equation 2.6.[7]

$$A(q, K, V) = \sum_i \alpha_i * v_i = \sum_i p(a(k_i, q)) * v_i \tag{2.6}$$

In summary, the value is amplified or reduced by the attention $\boldsymbol{\alpha}$. In the context of computer vision, the value typically is an image or contains similar visual data. Each element v_i could be a pixel in this example. Thus, each attention value highlights or hides individual pixels or spatial regions in the visual data.[7]

Importantly, the key and query information which is used to calculate the attention can come from various sources and can also stem from the spatial data. Further, the calculations of function **a** and **p** can be much more complicated and contain a variety of weights.

As this form is only the most general description of attention, a lot of detail is left to the specific implementation. This chapter will give an overview of attention categories and will discuss a number of specific implementations relevant for this thesis.

2.4.1 *Taxonomy*

An important part of attention mechanisms is not only how the attention is calculated, but also how and where it is applied. To differentiate these For this thesis, the taxonomy proposed by Chaudhari et al.[7] is used. Importantly, each category is not mutually exclusive and a specific implementation of attention may fall into more than one category. Also, not all types may directly apply for the task of video classification. See table 2.2.
Typically, when working with sequential data, only a single input sequence is considered at a time. If the output is also a sequence, Chaudhari et al. refer to this as **distinctive attention**. This type of attention can for example be found in machine translation tasks, where an input sentence needs to be translated into an output sentence.

Category	Type
Number of Sequences	distinctive, co-attention, self
Number of Abstraction Levels	single-level, multi-level
Number of Positions	soft/global, hard, local
Number of Representations	multi-representational, multidimensional

Table 2.2: Taxonomy of attention after Chaudhari et al.[7]

In contrast, **co-attention** weights apply to multiple input sequences at the same time and try to capture interactions between the inputs. For example, in the field of visual question answering multiple questions may be regarded simultaneously to detect context clues.

Lastly, **self-attention** concerns tasks like classification. Here, only the input is a sequence. Self-attention aims to capture relations of elements within the input sequence. This approach can be useful in natural language processing (NLP) applications, as words like verbs and nouns may or may not be in direct relation to each other.

When considering the number of abstraction levels, an attention mechanism may have a **single level** or **multiple levels**. Good examples for multi-level attention are visual attention modules, which can be inserted into various stages of a convolutional neural network.[29][49][48][66][69] As each layer of a CNN works with different abstraction levels of the input data, each attention module may capture different properties at each level.

In the third category, **hard** and **soft attention** represent two opposing approaches to attention. Hard attention aims to sample a subset of the input data for the network to process. This approach can reduce the computational cost, as the amount of data is reduced. At the same time, extracting a window from the input data results in a non-differentiable network which can not easily be trained with backpropagation[73].

In contrast, soft attention (also called **global attention**) can be envisioned as a mask of weights which is applied to the input. High values in the mask will highlight points of interests, while values close to zero can effectively hide parts of the data. The necessary weights can easily be learned through backpropagation, but in turn require more computations to calculate.

Finally, **local attention** can be regarded as a compromise between soft and hard attention. Here, the attention layer only considers a small window within the input data for its soft attention mask. Thus, similar to hard attention, only a section of the data is considered at a time. In comparison to the global soft attention mask, much fewer computations need to be performed.

In the realm of natural language processing, feature representations like word-embeddings are commonly used. Numerous embeddings exist, and each may capture different properties of the original input. **Multi-representational attention** may be used with two or more representations of the input data to combine and evaluated them. Similarly, a word

embedding typically is a vector and **multidimensional attention** can be used to weigh the importance of the individual dimensions of the embedding.

In the following the focus is on visual attention approaches which can be used for human action classification.

2.4.2 *Learn to pay Attention*

In 2018 Jetley et al. proposed a simple and modular attention unit which is intended to be inserted into many 2D-CNN architectures.[29] As a shorthand, it will be referred to as L2PA module. Their approach aims to correlate the high-order visual information contained in the last layer of the CNN with the more local information contained in the intermediate layers of the network. They denote these component as the global descriptor **g** and the local descriptor **L**, respectively.[29]

$$L = \begin{bmatrix} l_{1,1} & l_{1,2} & \dots & l_{1,j} \\ l_{2,1} & l_{2,2} & \dots & l_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ l_{i,1} & l_{i,2} & \dots & l_{i,j} \end{bmatrix} \tag{2.7}$$

Here **l** is a tensor with the spatial dimensions i and j and a number of channels. The specific dimensions of the values of the local descriptor depend on what layer it is extracted from. **g** is a vector and has a size of 512 in the following example. Figure 2.9 illustrates this.[29]

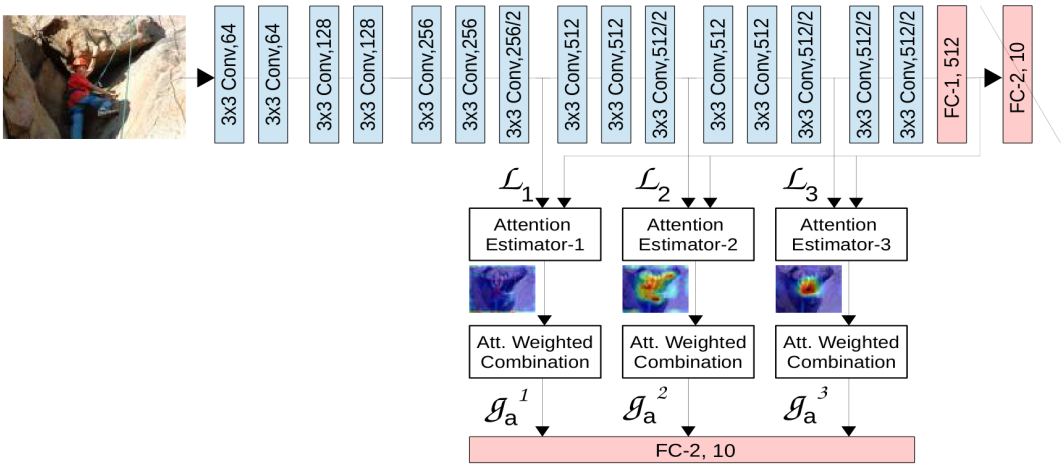


Figure 2.9: "Learn to Pay Attention"-modules applied to the VGG16 architecture. Image source: “Learn to pay attention” by Jetley et al.

To calculate the attention first the compatibility score **c** is computed. This is the dot product of a trainable weight vector **u** and the sum of each spatial entry **L** and the global descriptor. The size of **u** is chosen appropriate to the dimensions of **L** and **g**.

$$c_{i,j} = \langle u, l_{i,j} + g \rangle \tag{2.8}$$

Importantly, this calculation demands the number of channels of **L** and the number of elements in **g** to be equal. Should the numbers differ, a dense layer is inserted to adjust the size of **g**.[29]
After calculating the compatibility scores, the attention mask is derived by applying the softmax function to the result. During implementation, careful attention should be paid that this softmax function is applied over both spatial dimensions of **c**.[29]

$$a_{i,j} = \frac{exp(c_{i,j})}{\sum_i \sum_j exp(c_{i,j})} \tag{2.9}$$

The final attention g_a has the same spatial dimensions as its associated local descriptor.[29]

$$g_a = \begin{bmatrix} a_{1,1} & \dots & a_{1,j} \\ \vdots & \ddots & \vdots \\ a_{i,1} & \dots & a_{i,j} \end{bmatrix} \tag{2.10}$$

This attention module can be inserted at an arbitrary position in the base network. Although the authors note, the module requires fairly mature abstractions in the local descriptor. Therefor, they recommend inserting their module at the later stages of the network. Furthermore, as depicted, multiple attention modules may be used. For this approach, the authors propose two combination strategies. In their first approach, they concatenate the attentions $g_a^1 ... g_a^n$ and finalize the model with a fully connected layer. In their second approach, separate classifies are used by attaching individual dense layers to each result. Afterwards, these predictions are combined by averaging. In their findings, the authors find the concatenation approach to yield slightly better results.[29]

Jetley et al. test their attention mechanism on the VGG16 architecture[55] and use the 32x32 CIFAR-10/CIFAR-100 dataset[37] and a 80x80 version of the CUB-200-2011 dataset[65] to verify their results.

2.4.3 *Attention-Gated Network*

Attention-gated networks were proposed by, Schlemper et al. and can be understood as an extension and generalization of the mechanism of Jetley et al.[29]. They contribute three central improvements:[49][48]

- First they extend the core mechanism, thereby addressing the bottleneck described in the previous chapter.

- They propose a soft-attention alternative to the hard-attention used before.
- Finally, they propose a grid attention mechanism, which aims to evaluate localized information in images more finely.

The crucial change of the attention mechanism is realized by expanding the compatibility score equation 2.8 to the following.[49][48]

$$c_{i,j} = W_\Psi \sigma(W_l l_{i,j} + W_g g + b_g) + b_\Psi \tag{2.11}$$

Here $\boldsymbol{W_\Psi}$, $\boldsymbol{W_l}$ and $\boldsymbol{W_g}$ are learnable weights, while $\boldsymbol{b_g}$ and $\boldsymbol{b_\Psi}$ are learnable biases. Implementation-wise, this can easily be realized with 1D-CNN layers with a 1x kernel and a dense layer. In this case, $\boldsymbol{W_g}$ and $\boldsymbol{b_g}$ are part of a dense layer while $\boldsymbol{W_l}$ is part of a 1D-CNN. The final parameters $\boldsymbol{W_\Psi}$ and $\boldsymbol{b_\Psi}$ are part of a second 1D-CNN.

Another advantage of this approach is the introduction of the internal channel count $\boldsymbol{C_{int}}$ to describe the number of neurons of the dense layer and the number of filters in the 1D-CNN layers. Using this hyperparameter, the total number of trainable parameters of the module can flexibly be scaled up or down if necessary. If not otherwise specified, $\boldsymbol{C_{int}} = \mathbf{1}$ is assumed.[49][48]

Concerning the implementation of soft-attention, the authors firstly notice the application of the default softmax function (see 2.9) to be too sparse. Instead, they opt for using the following alternative, which has similar properties but does not limit the sum of the attention mask to be one.[49][48]

$$a_{i,j} = \frac{c_{i,j} - c_{min}}{\sum_i \sum_j (c_{i,j} - c_{min})} \tag{2.12}$$

Still, when utilizing hard-attention, the predictions are directly made from the generated attention. Schlemper et al. instead, propose to use the attention as a soft mask. They element-wise multiply the attention mask $\boldsymbol{g_a}$ with the original local descriptor $\boldsymbol{L}$. This way, the attention is able to highlight regions of interest in the spatial data while also concealing background noise.[49][48]

Last but not least, the authors notice the use of the global descriptor $\boldsymbol{g}$ as a single vector ignores a lot of the more localized spatial information. As an alternative, they propose a grid-based processing of this data. For this, they extract the information of the last 2D-CNN layer before the final pooling is performed. This yields a new tensor $\boldsymbol{g}$ with few spatial dimensions (e.g. 7x7) and a high number of channels (e.g. 1024). Each spatial position of $\boldsymbol{g}$ is used as a separate 1024-size vector when calculating the compatibility score and is applied to the respective spatial positions in $\boldsymbol{L}$. As a consequence, when choosing the layer to extract $\boldsymbol{L}$ from, one has to make sure the number of spatial positions is a multiple of the spatial positions of $\boldsymbol{g}$. For example, $\boldsymbol{g}$ being of size 7x7 would necessitate $\boldsymbol{L}$ to be of size 14x14, 21x21, 28x28 ... and so on. During the implementation of equation 2.11 one can simply use another 1D-CNN layer with $\boldsymbol{C_{int}}$ filters instead of a

dense layer to process the global descriptor.[49][48]

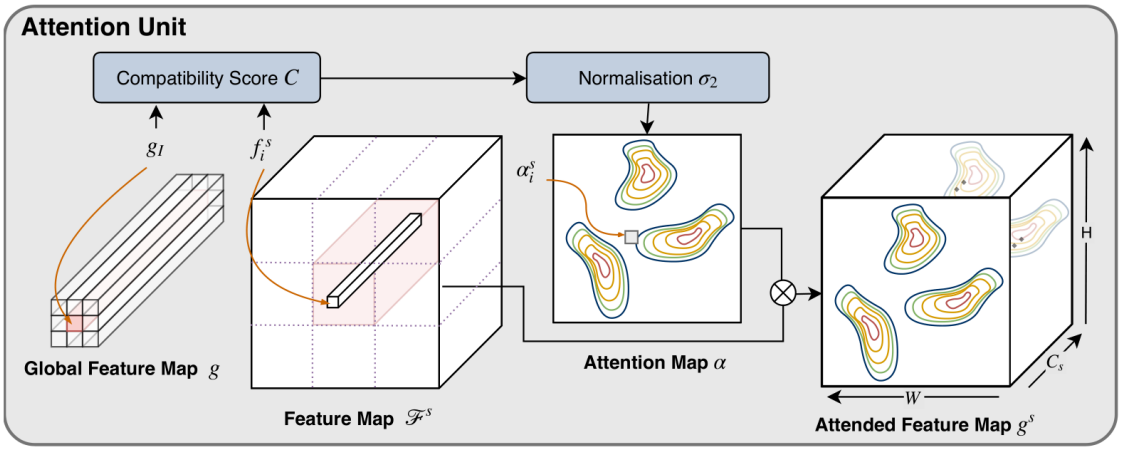


Figure 2.10: Gated attention unit. Here $\boldsymbol{g}$ has the spatial dimensions 3x3 and local features (here $\boldsymbol{F}$) are subdivided accordingly into a grid. Image source: "Attention-gated networks for improving ultrasound scan plane detection" by Schlemper et al.

Schlemper et al. verify their mechanism on the SonoNet[3] architecture. As dataset, they used a total of 2694 images of fetal ultrasounds with a final resolution of 208x272[48].

2.4.4 *Residual Attention Network*

In their attention mechanism, Jetley et al. and Schlemper et al. try to correlate local and global features to produces attentive features for the class prediction. Wang et al. propose an entirely different approach. Instead, they suggest a fully end-to-end trainable soft attention mask which can be inserted into residual neural network architectures (see section 2.2.3). In their design, the attention module splits the network path into a trunk branch and a mask branch. The trunk path contains regular residual layers as part of the usual network setup and is thus trivial. The attention itself is calculated in the mask branch. The mask branch first increases the receptive field by performing aggressive down-sampling using pooling layers. This way, the mask branch intends to capture global features in the data rather than local ones. Each down-sampling step is accompanied by a number of residual layers to capture the relevant information. In the second part of the mask branch, interpolation is performed to reproduce the original spatial dimensions. Each interpolation step is also accompanied by further residual layers, and an equal number of pooling and interpolation steps are performed.[66]

Lastly, the authors also introduce another shortcut connection before every pooling layer and after every interpolation layer. This shortcut connection is also parameterized by using a residual layer.[66]

The soft attention mask branch produces a tensor with the same number of spatial dimensions and channels as the input. As a final operation of the mask branch, a so-called attention function is applied. This function is essentially an activation function, like a sigmoid function.[66]

To merge the mask branch with the trunk branch, the authors employ a technique they name attention residual learning. They note if the simple element-wise product were to be performed, the attention module could potentially break desirable properties within the trunk branch. Using multiple attention modules with consequently lead to a degradation of the of features within the deep layers and a drop of the overall performance of the model. Instead, they propose the following function:[66]

$$H_{i,c}(x) = T_{i,c}(x) + (T_{i,c}(x) * M_{i,c}(x)) = T_{i,c}(x) * (1 + M_{i,c}(x)) \tag{2.13}$$

Where $\boldsymbol{H}$, $\boldsymbol{M}$ and $\boldsymbol{T}$ are the outputs of the attention module, mask branch and trunk branch respectively. The operations are performed element-wise for all spatial locations $\boldsymbol{i}$ and channels $\boldsymbol{c}$.

Using this function in the worst case, the output of the mask branch can default to zero and the attention module will produce the trunk output. This way, the authors argue, a residual network with attention modules will always perform at least as good as it would without them.

Figure 2.11 illustrates the structure of the entire attention module within a proposed residual neural network architecture.[66]

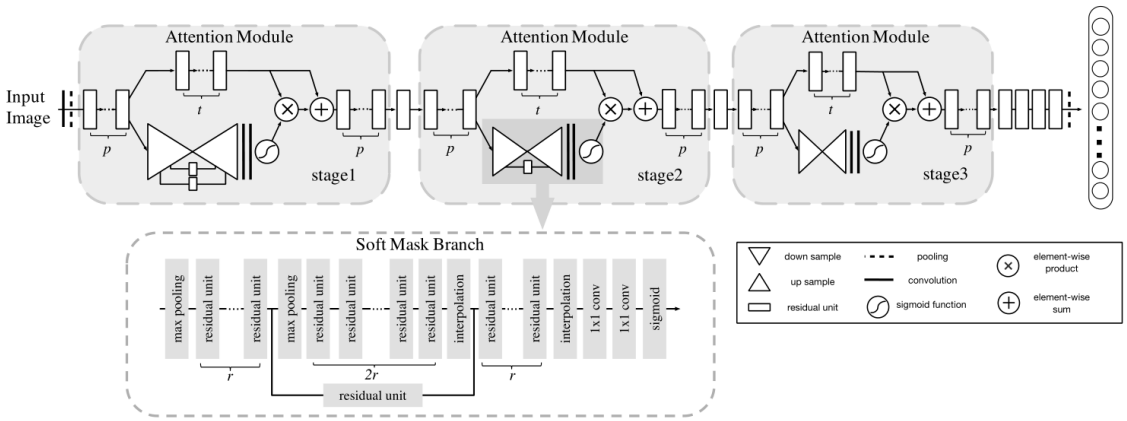


Figure 2.11: Overview of the residual attention network architecture. Image source: "Residual attention network for image classification" by Wang et al.

With this setup, the attention modules function both as feature selectors and noise suppressors. High values in the mask branch highlight regions of interest, while low values in the mask branch mask background information.[66]

Considering the implementation of this module, a few things have to be considered. First, the authors recommend using multiple attention modules to best capture the incremental nature of residual learning. Because the mask branch is intended to capture global properties rather than local ones, at the start of the network more down-sampling has to be performed. As the network progresses and the spatial dimensions decrease, the amount of down-sampling necessary within the attention modules also decreases. As every down-sampling and up-sampling is also accompanied by an additional short-

cut path, the number of shortcuts can also be used to measure the size of the receptive field of the module. In their proposed architecture Wang et al. use two, one and zero shortcuts for their three attention modules respectively. Also, every pooling step halves the spatial dimensions by two. Therefor, when inserting an attention module, the spatial dimensions of the data need to be divisible by a respective power of two.[66]

For further customization, the authors introduce a few hyperparameters and alternatives for their modules. First, as previously described, each attention module uses an attention function. The authors suggest three variants for this function.[66]

- **Mixed attention.** Here, the simple logistic sigmoid activation is used. The authors consider this the recommended default variant.
- **Channel attention.** The L2-norm is used.
- **Spatial attention.** A channel-wise z-score standardization is performed. Afterwards, the sigmoid activation is used.

Additionally, as can be seen in figure 2.11, the authors use the hyperparameters $\boldsymbol{p}$, $\boldsymbol{t}$, and $\boldsymbol{r}$ to parameterize their modules. Each variable corresponds to the number of residual layers padding the attention module, within the trunk branch and within the mask branch, respectively. The default value for $\boldsymbol{p}$ and $\boldsymbol{r}$ is one, and the default value for $\boldsymbol{t}$ is two. Finally, the number of shortcuts as described above can also be considered a hyperparameter.[66]

Wang et al. use the CIFAR-10, CIFAR-100[37] and ImageNet[11] datasets to validate their proposals. As underlying residual network architectures, they use classic ResNets[24] (see section 2.2.3) and the more advanced ResNeXt[72] and Inception-ResNet[58] architectures.[66]

2.4.5 *Convolutional Block Attention Module*

The convolutional block attention module (CBAM) was proposed by Woo et al. and follow a modularized approach similar to the previously discussed residual attention modules. Further, the authors note their convolutional block attention module to be an extension of Squeeze-and-Excitation Networks (SENets)[28]. While SENets "use global average-pooled features to compute channel-wise attention"[69], CBAMs also use max-pooled features and additionally apply this technique to spatial information. Figure 2.12 shows the structure of a convolutional block attention module. Here, $\otimes$ denotes the element-wise product.[69]

The channel attention module intends to weight the importance of different features contained in the channels. For this, the global average-pooling and global-max pooling is collected from the original feature tensor $\boldsymbol{F}$, resulting in two vectors. The size of both vectors is the channel count of $\boldsymbol{F}$. The authors deem average-pooling alone to be an

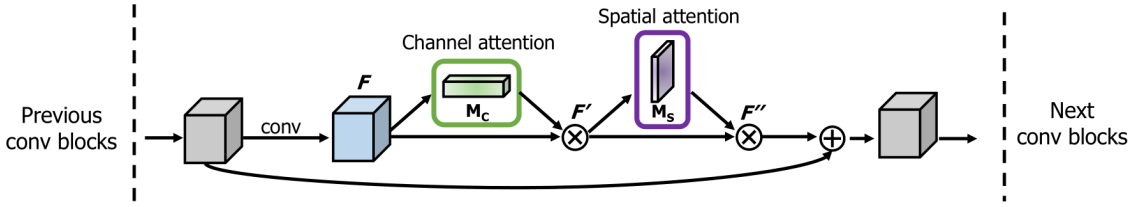


Figure 2.12: Convolutional Block Attention Module applied to convolutional features $\boldsymbol{F}$. Image source: "Cbam: Convolutional block attention module" by Woo et al.

insufficient representation of the original features. They argue the max-pooling features contain a different aspect of the data and are therefor valuable to include.[69]

To process both vectors, a simple multilayer perceptron (MLP) with a single hidden layer is used. Its size of the output layer is equal to the input layer. The number of neurons of the hidden layer is controlled by the reduction-ratio hyperparameter $\boldsymbol{r}$. For instance, a value of $\boldsymbol{r = 2.0}$ would result in a hidden layer half the size of the input layer and output layer. The authors suggest $\boldsymbol{r = 16.0}$ as default value. Although the multilayer perceptron is applied separately to each input, the weight-parameters shared. The hidden layer uses the Rectified Linear Unit (ReLU) activation function.[69]

Finally, the both outputs of the multilayer perceptron are added, and the sigmoid function is applied to the sum. Equation 2.14 concisely shows the operation done in the channel attention module.[69]

$$M_c(F) = \sigma(MLP(GlobalAveragePooling(F)) + MLP(GlobalMaxPooling(F))) \quad (2.14)$$

As previously seen, the resulting channel attention map $\boldsymbol{M_c}$ is then applied to $\boldsymbol{F}$ by element-wise multiplying it with each spatial position to produce $\boldsymbol{F'}$.

The spatial attention module intends to capture inter-spatial information within the features. Mirroring the channel attention module first the channel-wise average-pooling and max-pooling are performed. On a technical level, these operations are different from the typical pooling operations performed in a neural network. More adeptly, they can be described as mean-reduction or max-reduction along the channel-dimension. This results with a tensor with the original spatial dimensions and two channels.

To this tensor, a convolutional layer with a single filter is applied. The kernel of this operation is described by the hyperparameter $\boldsymbol{k}$. The authors suggest the default value of $\boldsymbol{k = 7x7}$. Equation 2.15 summarizes all steps taken in the spatial attention module.[69]

$$M_s(F) = \sigma(ConvLayer^{k=7x7}_{filters=1}(concatenate(ReduceAvg(F'); ReduceMean(F')))) \quad (2.15)$$

Just as previously, the resulting spatial attention $\boldsymbol{M_s(F)}$ by element-wise multiplying it with each channel of $\boldsymbol{F'}$ to produce $\boldsymbol{F''}$. $\boldsymbol{F''}$ in turn, is added to the original feature tensor $\boldsymbol{F}$ to produce the final attended features.

Similarly to the residual attention module discussed beforehand, CBAM effectively also uses the benefits of what Wang et al. describe as residual attention learning. The attention is added to the original features the same way as seen in equation 2.13. In the worst case, the convolutional block attention module can default to zero, leaving the features unchanged. Thus, one can expect a network with additional convolutional block attention modules to perform at least as good as one without them. Furthermore, each module is lightweight, having only relatively few trainable weights.[69]

Unlike the attention mechanisms previously discussed, convolutional block attention modules do not produce attention which can readily be visualized. Instead, the authors use class activation mapping (CAM) to visualize their results. This technique is discussed in detail in section 2.5.[69]

Specifically, the authors use Grad-CAM[50] to demonstrate that convolutional block attention modules result in the network focusing better on the intended features in images.[69]

Woo et al. adapt the ResNet[24], ResNeXt[72], WideResNet[76] and MobileNet[27] architectures for their classification experiments. As dataset, they use ImageNet[11]. In their results, they consistently outperform the respective baseline architectures, while the total number of network parameters and computation operations remain largely unaffected.

2.5 POST-HOC ATTENTION

Post-Hoc attention methods differs from the attention discussed so far, as they do not require a module or learnable weights. Instead, they aim to produce visual explanations from any trained neural network without any prerequisites.

Gradient-weighted class activation mapping (Grad-CAM) is one of these techniques and extends regular class activation mapping[82]. It follows these four steps to calculate a heat map for any convolutional layer of a neural network with an input:[51][50]

1. Forward-propagate through the network and save the activations for the convolutional layer of interest.
2. At the output layer, set the gradients to zero, except for the desired class, which is set to one.
3. Calculate the gradients for the layer of interest, by back-propagating through the network.
4. Multiply the class activation scores and gradients for the layer.

The resulting score is the Grad-CAM heat map for the layer.

This technique has a number of major advantages. Firstly, the class activation can be flexibly set in step 2. of the algorithm. This allows to inspect the class activation mapping for the prediction of the neural network, but also allows investigating if the network detects any other class of.

Secondly, Grad-CAM can be combined with guided backpropagation to produce a more fine-grained class activation mapping. This technique is called Guided Grad-CAM.[50]

And finally, this technique is not limited to image classification and can easily be adopted for other computer vision task like image captioning or visual question answering.[50]

2.6 VIDEO DATASETS

When discussing datasets for human action recognition, one first has to be mindful of the task-specific challenges addressed in chapter 2.1. Each individual dataset may be affected by these fundamental problems to a different degree, depending on its properties.

In their analysis of the subject, Kong and Fu [35] examine a number of popular datasets. Beyond the obvious features of a classification dataset, namely the number of records and the number of classes, they also identify further basic properties human action recognition datasets can commonly be described by:

If the data is of a **controlled or uncontrolled environment** depends on where the video data has been recorded. In a controlled environment, there typically is a single, fixed recording position. The camera itself does not move, the illumination conditions are stable, and the background is fixed. In an uncontrolled environment, none of these conditions apply.

The **number of views or angels** a scene is recorded from. More views may yield additional information which would otherwise be obscured or hidden.

Most video-datasets are **RGB**-datasets, containing only information which can typically recorded with cameras. More recently, **RGB-D**-datasets have been record, which contain additional distance information.

In some datasets, the **Number of subjects** is known. This refers to the number of humans involved in recording the video data.

The information above is useful for a concise summary of a dataset. Yet there a many more features and considerations which are important, but can not be evaluated at a glance:

- Some datasets may contain inaccurate or **noisy labels**. For example, for the YouTube-8M dataset the authors evaluated a variety of metadata, user input like anchor texts and comments as well as other information to automatically generate labels [1].
- The **video formats, resolutions and lengths**, as well as the **uniformity** of those features may be important details during implementation.
- Human actions can be subdivided into **group actions and individual actions**. A dataset may contain both or exclusively one of the types.
- A dataset may be **annotated with additional labels**. Depending on what the dataset is used for, information like this can be crucial.

- The original **source** and the provider of the dataset may play a role for various considerations. This can firstly be noteworthy when estimating the bias or other shortcomings of the data. Secondly, license terms or information privacy considerations may also be relevant when evaluating a dataset for its intended use.
- The **predictability stage** describes how quickly and when an action can be identified. Certain actions like playing billiard can be identified instantly or early, while other actions like throwing can be predicted late [36].
- The **specificity** of a dataset may be an important property to consider. Some datasets are concerned with highly specific parts of human actions, like gesture detection. Other datasets are more generic, covering broader subjects like sports. Finally, a heterogeneous dataset may contain action types from a broad spectrum of situations [6].

Lastly, one may also want to consider other meta-properties which are difficult to determinate. For example, it may play an important role in how well a dataset is researched. A more widely analyzed dataset may have more reference implementations, benchmarks and analysis, which in turn can help to verify new research and approaches.

2.6.1 *HMDB51*

The Human Motion DataBase (HMDB51) contains almost 7000 video clips manually labeled into 51 categories and was the largest action dataset at time of release in 2011.[39] For the creation of the datasets, students were asked to find video clips of humans performing single non-ambiguous actions. These clips were chosen from movies, public databases and internet sources like YouTube.

The authors subdivide the 51 classes into another five groups: 1) Facial actions like *smiling*; 2) Facial action with objects like *smoking*; 3) Body movements like *jumping*, 4) Body movements with objects like *kicking a ball* and 5) body movements with human interaction like *shaking hands*. Practically this and other information is contained in the additional meta-data annotations the authors provide:

- The **visible body parts** describes the visibility of the *head*, the *upper body*, the *lower body* or the *full body*.
- The **camera motion** describes if the camera is *moving* or *static*.
- The **relative view point** describes if the actors are seen from the *front*, *back*, *left* or *right*.
- The **number of people** describes if *one*, *two*, or *multiple* people are involved in the action.
- The **video quality** described the presence of motion blur and compression artifacts and is either *low*, *medium* or *good*.

Due to the varying data sources, the original videos differ in formats, resolutions and frame rates. The data was scaled by the authors to a height of 240 pixels while maintaining the original aspect ratio. The clips therefore vary in frame width. The frame rate

was converted to 30 frames per second. Due to 59.9 % of the videos containing camera motion, the authors assumed video-stabilization to be a common pre-processing step and additionally provide a stabilized version of the same data.

An important note when working with HMDB51 is that individual clips were often created from the same video source. As the authors point out, it is undesirably to use clips from the same source in training and testing, as this would lead to uncharacteristically good test scores. Furthermore, the various additional characteristics and annotations should be distributed in a proportional fashion between the different sets. Therefore, dividing the data in training and test sets is a non-trivial problem. Thus, the authors provide three different splits which were hand-selected from randomly generated spits while respecting the aforementioned criteria.

2.6.2 *UCF101*

The University of Central Florida UCF101 dataset[56] was conceived in response to a number of perceived problems of video datasets of the time. The UCF101 is a superset of its predecessor UCF Sports, UCF11 and UCF50. Many of which had relatively few samples, records or were recorded in controlled environments. In distinction, UCF101 contains 13320 video clips across 101 action classes, contains clips from an uncontrolled environment and was the largest video dataset at the time. The video data was manually downloaded and labeled from YouTube. Similarly to the HMDB51 dataset, the authors also subdivided UCF101 into 5 groups: 1)Human-Object Interaction like *biking*; 2) Body-Motion Only like *push-Ups*; 3) Human-Human Interaction like *hair cutting*; 4) Playing Musical Instruments like *drumming* and 5) Sports like *fencing*.

Unlike HMDB51, UCF101 does not have additional meta-data annotations. However, the authors note their dataset to be challenging, as it contains poor lighting, cluttered backgrounds and severe camera motion.

All videos have a resolution of 320x240 and a frame rate of 25 frames per second. The clips vary in length, with an average length of 7.21 seconds. Some clips in the same group share common features, such as backgrounds and actors, and there are three different splits for action recognition provided. Additionally, a further three splits for action detection and space-time interest points (STIP) features can also be downloaded.

2.6.3 *Other Video Datasets*

This section addresses another few noteworthy datasets, which warrant discussion but can not be examined in detail.

The Sports-1M dataset consists of one million videos of 487 classes. A small amount of the data is annotated with multiple labels. The labels were automatically generated from metadata and may contain a small amount of inaccuracies and there may be a

small overlap between the training and test data.Karpathy et al.

YouTube-8M is another dataset curated from YouTube videos, as the name suggests, consisting of eight million videos. Instead of being a single-label classification dataset like the other video datasets examined so far, YouTube-8M is a multi-label classification dataset with a total of 4800 different entities.Abu-El-Haija et al.

The 20BN-something-something is noteworthy because of its unique focus on less specific actions. Rather than classifying specific actions like *archery*, the something-something dataset has classes like *picking [something] up*, where *[something]* can be any object. Otherwise, the dataset has over 100,000 videos ind 174 classes.[20]

2.7 IMAGE DATASETS

Image datasets are helpful when dealing with architectures like two-stream networks, which internally use 2D-CNN. Firstly, they allow training and testing the backbone architecture individually, before approaching video datasets. Secondly, pre-training may be used. This section covers a few of the most common image datasets, before addressing this topic.

The CIFAR-10 and CIFAR-100 contain 60000 images in 10 and 100 classes respectively. With a resolution of 32x32, it is a fairly small dataset, but is nevertheless commonly used.[37] Due to its small size, it can be used to quickly train and test image classification models.

The ImageNet dataset is arguable one of the most important image datasets due to its size and due to its role in pre-training.[22] The project is constantly growing and the largest image database.[11] The most used subset of the ImageNet data is the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset, containing 1,431,167 annotated images in 1000 classes.[46]

The Imagenette dataset[1] is a subset of ImageNet dataset with 10 easily identifiable classes. It is therefor well suited to quickly train and troubleshoot image classification models.

Finally, image datasets also play a small but important role in video classification. Pre-training models on image-datasets, and especially the ImageNet dataset[11], is a well established practice when using convolutional neural networks.[22]

This practice often helps image and video models to reach better scores on the target dataset. Although it should be noted that more recent research critical reevaluates pre-training and shows that although it helps speeding up early training, it does not

1 See https://github.com/fastai/imagenette.

generally help to improve the final prediction quality, unless the video data used is insufficiently in quantity.[22]

2.8 DATA PREPROCESSING

This section addresses two data preprocessing steps common to computer vision. The first topic of image data augmentation applies to image data and how it can be modified to increase the amount of usable training data and reduce the risk of overfitting.

The second topic, optical flow, describes a technique which allows detecting and extracting motion information from video data. This is a crucial preprocessing steps for two-stream networks and other architectures which rely on optical flow to process temporal information.

2.8.1 *Image Data Augmentation*

A common problem in computer vision are insufficient quality and quantity of the datasets, as both issues will usually result in overfitting. Image data augmentation encompasses a multitude of techniques to increase the number of usable training examples. The core of these techniques is to modify the original data while leaving the image recognizable. The resulting modified image can be used as another training example. Broadly these techniques can be categorized into geometric image manipulations, advanced image manipulations and deep learning based image manipulations.[53]

Basic geometric image manipulations consist but are not limited to flipping the image vertically or horizontally, cropping the image, rotating the image, shear-rotating the image, shifting the image, injecting noise or transforming the color space. Each method should be evaluated carefully for the dataset. For example, horizontal flipping may be useful for many image manipulation task, but may be detrimental when applied to a character recognition dataset like MNIST[12]. The value ranges for each transformation should also be selected carefully. For example, modifying the color space by adding too large values may leave the image unrecognizable.[53]

More advanced manipulations like applying a kernel filter to an image can yield various results, as seen in section 2.2.1. Other advanced techniques are combining images and randomly erasing parts of images. Both of these techniques may be counter-intuitive to understand, as they seem nonsensical from a human perspective. Yet they have been shown to improve the results when training a neural network with data augmented this way.[53]

Finally, a number of deep learning based techniques like adversarial training, GAN-based data augmentation, neural style transfer or meta learning exist.[53] However, these techniques can not be addressed here in detail, as they too advanced to fit the scope of this thesis.

2.8.2 *Optical Flow*

Optical flow describes the movement of objects in relation to an observer, or the apparent movements in case of a moving observer. Fundamentally, optical flow is calculated from two images and aims to capture the motion that occurred between these images.[26]

Optical flow can be most easily be described by addressing a single pixel. First, assume the exact positional change, both horizontally and vertically, of this pixel between the images is known, then this change in the spatial positions can be described as a vector. The resulting displacement vector contains the directions and magnitudes of this apparent movement and is called the optical flow.[16]

In the case of dense optical flow, this vector has been calculated for each pixel in the original images, resulting in a dense displacement field.[16]

The definition of optical flow depends on the assumption the exact positional change of individual pixels is known. Practically, this is not a case and this has to be estimated. Various methods to estimate optical flow and dense optical flow exist[4][16][44], which can not be addressed within the scope of this thesis. Instead, the TV-L1 optical flow[44] is used because it is readily available as part of the OpenCV library[2] and because it is commonly used in video classification with neural networks [5][57][32].

2.9 EVALUATION CRITERIA

When evaluating a classifier, there are a few important metrics to consider.

- **Top-1 accuracy**. How often the classifier predicts the correct class. Sometimes the top-1 misclassification score is used instead.
- **Top-5 accuracy**. How often the classifier has the correct class among its top five predictions. The counter-value is the top-5 misclassification score.
- **Training Time**. Adding more trainable weight parameters will also affect the overall training time. The difference can for instance be measured by the number of floating-point operations (FLOPs) involved in training the network.[69]
- **Explainability**. Specifically, when investigating attention mechanisms, the goal also is to improve how well the neural network can be diagnosed. Visual attention can help to show what a model focuses on. If the wrong objects or undesirable sections of a video are highlighted, the model maybe unsuitable or mistrained. This property may also be difficult to determine, as the examination has to be done by a human and can not be accurately judged by a metric. Also, evaluating more than a few records of a dataset may be time-consuming.

Lastly, when evaluating these metrics, it is important to perform cross-validation to ensure a high confidence in the produced values.

2 See https://docs.opencv.org/4.x/dc/d4d/classcv_1_1optflow_1_1DualTVL1OpticalFlow.html

3

RELATED WORK

So far, only subjects have been discussed which pertain directly to the subject of this thesis. Computer vision, artificial neural networks and the use of attention are extremely broad fields of research and many relevant topics could not be included due to the scope and time restraints of this work. This chapter intends to give an overview of other important developments and advancements, which may be relevant for interested readers. For the sake of structure, this chapter has been subdivided into related work concerning attention, other network architectures, different approaches to video classification in general and finally other relevant research which does not fit into the aforementioned categories.

3.1 ATTENTION

Attention itself is a broad and abstract concept and can be used in a multitude of ways. Attention mechanisms in artificial neural networks were arguably conceived by Bahdanau, Cho, and Bengio[2] and further popularized by Vaswani et al.[63]. These authors work on the problem of machine translation, and attention generally enjoys popularity in the domain of natural language processing.[17][7] Despite this, as shown in section 2.4 the concept can also be applied to computer vision.

So far, the topics discussed focused on image and video classification. Another important computer vision task is image detection and segmentation. In fact, the attention mechanism proposed by Jetley et al., the gated-attention mechanism[48] and the convolutional block attention module[69] were all also adopted and tested for this task. This

Another subject of interest is image synthesis and generation. For instance, attention can be used in generative adversarial networks (GAN). An example of this are Self-Attention Generative Adversarial Networks (SAGAN). [78]

Further, some applications overlap into multiple domains. Visual question answering and image or video captioning and description fall into both computer vision and natural language processing. xu2015show present one such attention model for the task of image captioning, with a hard attention approach and a CNN-LSTM architecture[73].

Natural language processing concerning speech also intersects with the domain of audio processing.

An entirely different field of research is the application of attention in graph models. Many modern tasks and problems can naturally be found as graph-problems. This includes a number of topics originating in computer science and software design, like the structure of the World Wide Web or social networks. Many real-world problems from the domains of bioinformatics, chemoinformatics, urban planning and most importantly due to recent events, epidemiology, can leverage graphs. Due to the complexity

of patterns and relationships that can occur in graphs, attention is an attractive possible solution to address these challenges.[40]

3.2 ARCHITECTURES

While some other image processing architectures have been mentioned, so far the focus lied on regular CNN and their extension of ResNets.[24] Because many video processing architectures use an image-model as backbone, it is important to be aware of more recent advancements in this field. In the following, some of the most important recent developments are chronologically summarized.

The GooGleNet architecture, also later known as Inception-V1, is based on a simple observation.[59] Ideally, when linearly stacking convolutional layers in a model, the kernel size should be selected depending on the size of the object of interest in the images. I.e. a classifier for dogs would need different kernel sizes whether the dog was photographed from a close or far angle. Accordingly, the authors proposed an inception module which contains several parallel CNN layers with kernel sizes 1x1, 3x3, 5x5 and a 3x3 max pooling operation. The resulting filters are concatenated at the end of each block. This allows the network to detect features with receptive fields of various sizes simultaneously. Additionally, to manage the number of necessary calculations the CNN layer of kernel size 3x3 and 5x5 were prepended with CNN layers of kernel size of 1x1, to limit the number of filters proactively.

This *split-transform-merge* strategy of dividing the data stream up with 1x1-convolutions to process them individually was later adopted by the authors of the ResNeXt architecture in a more extreme manner.[72] As a downside of the GooGleNet architecture, its individual inception blocks have a need to be carefully balanced for the stages of the network. In a ResNeXt block instead, there are numerous parallel paths consisting of uniform components. In each path the data is *split* with a 1x1-convolution, *transformed* by a 3x3-convolution and then *merged* with another 1x1-convolution. Most importantly, the outputs of each path are combined by addition instead of concatenation, to preserve the desirable properties seen in ResNets. As the name implies, ResNeXt also features an additional residual shortcut connection in each block.

The use of residual connections and other advancements were also incorporated into the architectures leading up to Inception-v4 and Inception-ResNet.[58] The central developments in these newer versions were the use of layer-level batch-normalization and the factorization of convolutional layers. For example, a 5x5-kernel convolutional layer can be factorized into two consecutive 3x3-kernel convolutional layers; a 3x3-kernel convolutional layer can be factorized into consecutive 3x1- and 1x3-kernel convolutional layer. Both factorization keep a similar ability to abstract information, but can do so much more efficiently with fewer necessary computation.

Concurrently, recent architectural research has also been focused on efficiency in addition to effectiveness. One such example is the MobileNet architecture developed by Howard et al. The authors noted, that convolutional layers are highly expensive in terms of computations, as the cost scales multiplicatively with the number of channels in the

previous layer and the number of filters of the layer. As an alternative, they propose depthwise convolutions, which establish a 1-to-1 relation between each input and output channel. With this approach, the number of calculations for each convolutional layer are divided by the number of channels. Subsequently, a simple so-called pointwise 1x1-convolution is used to process inter-channel relationships and adjust the number of channels if necessary. Using this approach, Howard et al. adapted the Inception-V3 architecture and were able to reproduce similar accuracy scores while using 8 to 9 times fewer calculations and weight parameters.[27]

Similarly Tan and Le also examined the effectiveness of network architecture design trends and particularly focused on the question of network scaling. Traditionally, increasing the network depth, i.e. number of layers, was the most common method to scale an architecture up to achieve better accuracy scores. Other scaling methods are the increase of width, i.e. the number of convolutional filters, and the increase of the resolution of the initial image. The authors note, that adjusting these three scaling methods has diminishing returns on accuracy gains. Due to this interdependence, focusing just on a single method is therefor inefficient. For example, one may intuitively understand, that increasing the input resolution also demands a larger receptive field to utilize this information. Consequently, Tan and Le propose a compound scaling method for depth, width and resolution and show how it can be used with the help of a resource-scaling coefficient to construct efficient networks. Using this method, they develop a family of networks called EfficientNets and compare their architecture to others like ResNet, Inception-V4 or ResNeXt and generally reproduce accuracy scores with a fraction of the weight parameters.[60]

Concerning other architectures for video classification, there are also more concepts and architectures not yet covered. As described in chapter 2, many video classification architectures indirectly use structural ideas and concepts from image-processing or directly use an image-processing network as a backbone component. Examples for this are the residual connections and two-stream networks, respectively. Another important idea is the inflation of 2D-convolutions into 3D-convolutionan to appropriate image-classification networks for the task of video classification. This technique has been demonstrated by carreira2017quo by inflating the Inception-V1 architecture trained on ImageNet.[5] The inflation was achieved by expanding the convolutional kernels of size NxN to size $NxNxN$ by repeating the weights. The weights values were subsequently divided by N to ensure the response of the kernel stays the same. The resulting network was used as a backbone in a two-stream architecture, resulting in a Two-Stream Inflated 3D-ConvNet (I3D).

In another more recent development, Stroud et al. hypothesize modern spatio-temporal network architectures should be able to abstract motion and should theoretically not require optical flow preprocessing to do so. Despite this, 3D-convolutional network architectures have consistently been shown to perform better with the addition of optical flow in research. The authors show 3D-convolutional networks would fundamentally able to calculate the optical flow information, but do not generally learn to do so during

regular training. Using this insight as a foundation, the authors develop a method of distillation, using a teacher-network conventionally trained on optical flow data. This teacher network is used to distill knowledge to a student network of the same architecture. The intermediate is goal is to reproduce the information available within the teacher network in the student network, which uses the original RGB-information as inputs. Subsequently, the resulting distilled 3D network (**D3D!**) requires no optical flow information during further training and inference.[57]

Last but not least, another important development in neural network architectures are transformers. They were introduced in 2017 by Vaswani et al. as an alternative to the LSTM architecture commonly used for NLP tasks.[63] Transformers are doubly interesting, because a) they allow processing sequential data in a parallel fashion without relying on recurrent connections and b) because they use attention as a core component in their architecture. Transformers are split into an encoder and decoder section. Both of the sections consist of repeating attention units and densely connected units, both of which also employ residual shortcuts and batch normalization. As attention function, Vaswani et al. propose the simple scaled dot-product or a so-called multi-head attention function, which conducts parallel scaled dot-product attention by linearly projecting the input values. Regarding the taxonomy introduced in section 2.4.1 this type of attention falls under self-attention.

A notable advancement are Bidirectional Encoder Presentations from Transformers (BERT).[13] [32] The BERT architecture addresses a few problems commonly found in natural language processing with self-supervised pretraining. For example, a network can be pretrained on a neutral text-corpus like Wikipedia and can later be used on the target problem. During training, this is achieved by masking individual words and sentences and having the network predict the masked information. As a bidirectional model, the sequences are evaluated from both directions. I.e. when processing the sentence *"opening a [bank] account"* the network can use the information of *opening a* and *account* to predict the word *bank*.

Since their initial conception, the use of transformers in computer vision has seen increased interest.[34] At the time of writing, a transformer architecture called "CoAtNet", combining convolutions and attentions, achieves state-of-the-art results on the ImageNet dataset.[10]

3.3 APPROACHES

So far, this thesis only discussed RGB-datasets with three color channels for red, green and blue. In part due to consumer available recording devices like the Kinect more research into evaluating depth information has been done.[71][35] Depth information is particularly interesting, as it easily allows detecting objects of interest and can be used to reducing background noise. This is intuitive, as depth information is invariant to lighting conditions and objects of interest are often much closer to the camera. But RGB-D-datasets also come with a new set of additional challenges. Firstly, they are even larger

than regular video datasets, making it an even bigger problem to process them. For this reason, many RGB-D datasets hold only a few hundred records.Zhang et al.[35][41] Additionally, depth-based action recognition may be more susceptible to further problems like viewpoint variations, biometric variations or occlusion of body parts.[41]

When discussing RGB-D datasets, a natural followup topic is skeleton-based action recognition. Depth data can be used to efficiently detect human body parts and joints to create a simplified 3-dimensional skeleton.[79] The detected joints or body parts can then be used as features for a shallow or deep model.[67] Skeleton-features can also be used in conjunction with regular depth information to increase the overall robustness of the predictor.[67]

3.4 OTHER RESEARCH

Another few topics warrant discussion but could not be neatly fitted into the previous categories.

As with all data-driven applications, the amount, quality and variety of the data are important factors for the final quality of any model. This is especially a problem with video datasets, as manual labeling of videos is much more time-consuming than labeling images. Common practices to address this problem is to pretrain a video model on an image-dataset like ImageNet or to incorporate other video-datasets during training. Yet the central problem still remains, despite the huge amounts of video data being available on the internet, only a tiny fraction is well-annotated and can readily be used to train models.

Accordingly, solutions to this problem have been researched. An obvious solution is to automatically generate annotations from available metadata, as it has been done for datasets like YouTube-8M[1]. As a downside, these automatic generation-methods themselves are not perfect and a certain amount of the data will be mislabeled. Further, this method is still static in nature. The dataset still has to be curated and published.

As an alternative, webly supervised learning aims to directly leverage data from online sources like search engines or image hosting websites.[14][8] This task, of course, has also many challenges. Search engines may be biased towards easily recognizable results with a low diversity. Image hosting websites instead may have noisy labels. Newer research to adapt this learning strategy for video recognition models also exist.[15]

Finally, the selection of frames has not been discussed in detail. Videos typically contain more frames than can be feasibly processed, and only a small subset is selected from the original video. So far, the frames have been uniformly sampled from the original video. As discussed in section 2.1, one of the problems of action recognition is the uneven amount of information contained in each frame. While most frames are redundant, some key frames may be crucial to recognize an action. Selecting the right frames can therefor affect the accuracy of a model positively. Selecting the unnecessary frames will negatively affect the overall performance of a model. Based on this observation,

Gowda, Rohrbach, and Sevilla-Lara propose a smart frame selection mechanism. They train two auxiliary networks to weight the importance of frames a) individually and b) across frames.[19] Using this method, they report improvements of the accuracies of the tested architectures while also being able to reduce the computational cost necessary by a factor of 4 to 10.

4

CONCEPTION

The research in chapter 2 suggests many attention mechanisms can potentially be used in video classifications models. Moving forward, there are three central questions this thesis aims to answer:

- Which attention mechanisms can be adapted for video classification?
- Which video classification architectures can benefit from attention?
- How can these adaptions be implemented?

Due to the broadness of the topic, finding exhaustive answers to these questions would require prohibitive amounts of research and empirical verification. Therefor, this work will limit itself to a number of promising attention mechanisms and architectures. This chapter describes the approach and the reasoning behind the decisions that were made.

4.1 APPROACH

As described, the above questions are too broad to be fully answered and need to be narrowed down. For this reason, this approach first subdivides the questions into a number of steps which can more easily be answered.

1. **Identifying suitable types of attention**. Not all types of attention can be adapted for the task of video classification. If too many candidates are found, a subset is selected for implementation.
2. **Identify suitable architectures for video classification**. As witch attention mechanisms, only a few architectures may be implemented within the scope of this thesis.
3. **Choosing a backbone architecture**. Typically, video classification architectures use an image classification network as backbone. Ideally, the backbone network should be compatible with most or all attention mechanisms and video classification architectures selected.
4. **Implementing image classification prototypes**. Ideally, since the backbone architectures are preferred to be compatible with the attention mechanisms, pure image classification models can be created as prototypes. This allows to test the implementations and verify their correctness before they are used within the video models. In many cases, this step can be viewed as a small replication study of the original research papers of the respective attention mechanism.
5. **Implementing video classification models**. Finally, the prototypes can be used in the actual video classification models as backbones.

6. **Producing benchmarks**. To verify the results, several benchmarks are produced. This can be done preliminarily for the prototypes and afterwards for the video models.

Using this approach, the goal is to show how attention can be used in video classification. Furthermore, the intent is to show the efficacy and advantages of attention specifically for a few selected architectures.

4.2 SELECTED ATTENTION MECHANISMS

To first determine which types of attention are suitable, it is helpful to look back at table 2.2.

As each video only consists of a single sequence of frames, there are few options when it comes to the number of sequences. Distinctive attention is used in sequence to sequence tasks. The usage of co-attention is theoretically possible, but it is practically prohibitively difficult to deploy attention to capture dependencies between different videos. Processing videos is already difficult due to the large amount of data involved. Processing multiple videos simultaneously to use co-attention would only exacerbate this issue. Consequently, self-attention is the primary type of attention which will be investigated.

Concerning the number of abstraction layers, the situation is less restricted. Convolutional neural networks lend themselves well to multi-level attention. Using attention at different stages of a CNN may capture different abstraction levels of the data. Using modularized attention at different levels therefor seems like a promising approach.

The number of positions, again, leaves fewer choices. Hard attention typically makes it difficult to train the network, as the model function becomes non-differentiable. This is highly undesirable and poses further problems if the model is to be used as a backbone within another architecture. Therefore, soft types of attention are more suitable, as they leave the network end-to-end trainable and are less intrusive.

Finally, as a category, the number of representations generally only applies to language models. In computer vision, embeddings are much less common. Even though a CNN may be used to embed an image to produce a feature vector, it is highly untypical to use multiple such representations simultaneously. Similarly, one may regard optical flow information as another representation of the original video data. But in this case also the interdependencies between the original data and the optical flow data are of little interest. In a typical two-stream architecture, both streams are processed independently. Inserting this type of attention would require large adjustments of the architecture.

In summary, the best candidates are soft self-attention mechanisms. Additionally, modularized mechanisms may be particularly interesting, as they can easily be inserted into existing architectures and can be used at multiple levels of a CNN to leverage different abstraction levels.

Another important criteria is the visualization of the attention. One of the central goals of attention is helping to make neural networks easier to interpret. This is especially true in the field of computer vision. It is therefor desirable to implement types of attention which result in visually evaluable results. Based on this, the following attention mechanisms were chosen. See table 4.1.

Attention mechanism	Categories	Modular?	Visual?
L2PA	Self-attention, multi-level, soft	Yes	Yes
Gated grid attention	Self-attention, multi-level, soft	Yes	Yes
Residual attention	Self-attention, multi-level, soft	Yes	Yes
CBAM	Self-attention, multi-Level, soft	Yes	No

Table 4.1: Selected attention mechanisms.

As a downside of this selection, it has to be mentioned all these attention mechanisms are designed for 2D-CNN and therefor work exclusively on spatial information. They are unable to capture any temporal dependencies directly. Arguably, this downside is mitigated when optical flow information is used, which allows processing temporal information with spatial convolutions. Further, it can be argued the larger amount of information in video sequences is visual, rather than temporal. Following this argumentation, it is reasonable to focus on visual attention.

4.3 SELECTED VIDEO ARCHITECTURES

First and foremost, simple 2-dimensional CNN can be excluded as a naive approach to video classification. Depending on the fusion strategy, they have no effective way of capturing temporal information. And even with more advanced fusion strategies, they are very limited in this regard.

Instead, the focus lies on the following four architecture archetypes:

- **3-dimensional CNN** and variations like (2+1)D-CNN. A backbone architecture can not directly be integrated, but often the layout of 3D-CNN is inspired by image classification models.
- **Long short-term memory networks** with a 2D-CNN used as backbone.
- **Transformer networks**.
- **Two-stream networks** also with a 2D-CNN used as backbone.

3D-CNN and (2+1)D-CNN use convolutions to capture both spatial and temporal information. Unfortunately, this also makes it more difficult to adapt them with the attention mechanisms chosen. All the attention mechanisms are primarily designed for 2-dimensional convolutions. It is an open question if the mechanisms are even suitable for adaption into 3-dimensional convolutions.

Long short-term memory networks typically use a 2D-CNN to extract spatial features from individual frames and subsequently use LSTM units to process temporal information. This setup works well with visual attention modules, which can directly be inserted into the backbone architecture with relatively little effort.

Transformers are an interesting and relatively new architectural archetype and can be seen as a successor of long short-term memory networks. Especially in the natural language processing domain, they have replaced LSTM as state-of-the-art solutions for many tasks. Transformers also use self-attention as a fundamental building block within their structure. For this research, however, this feature poses a problem. Directly utilizing different attention modules may conflict with the attention already in use, and the attention modules chosen are primarily intended for 2D-CNN architectures and may not be compatible with transformers.

Two-stream architectures, can directly use two separate 2D-CNN models as backbone for spatial and temporal information. This makes them well-suited for adaption with the attention modules chosen. The usage of optical flow data within two-stream networks is a particularly interesting feature, as it allows using the attention modules discussed not only for spatial information but also for temporal data. This also addresses the major drawbacks of the attention modules chosen, being unable to process temporal information directly. Furthermore, due to the prevalence of optical flow in the field of computer vision, this poses another interesting research question. Are visual attention mechanisms suited to be used for optical flow data? Due to these circumstances, two-stream networks are of particular interest for this research.

In review of the suitability of the architectures discussed and the constraints and scope of this thesis, the decision was made to focus on the two-stream architecture archetype. This allows not only to research the overall general compatibility of visual attention modules with video data, but may also particularly provide insight into the question of their usage in conjunction with optical flow.

4.4 BACKBONE ARCHITECTURE

Both the attention modules and the video classification architecture chosen rely on a 2D-CNN architecture. Hence, choosing one or multiple backbone models is also an important decision. A few major considerations were made for this decision.

- **Effectiveness**. If reconcilable with the other requirements, the architecture should yield as good results as possible. Thanks to the prevalence of the ImageNet dataset, the effectiveness of image classification architectures can relatively well be compared by benchmarks on this dataset.
- **Efficiency**. Due to resources constraints, the efficiency of the chosen backbone architecture is extremely important. More lightweight architectures allow for faster

training and enable training with fewer resources. Therefore, the architecture should be as efficient as possible.

- **Adaptability**. To be able to be used with the attention modules and as a backbone model, the architecture should be relatively simple and suitable for adaption. More complex architectures may be more likely to be incompatible. However, the assessment of this property may be difficult.
- **Research status.** It is preferable to use well-researched and well-established architectures. Architectures with less research may be less reliable and have fewer resources and reference-implementations available.

Based on these criteria, the following architectures were initially taken into account. All the architectures were considered sufficiently well-researched. See table 4.2.

Architecture	Parameters	Top-1 Accuracy	Complexity assessment
VGG16 [55]	138.3 Million	0.713	Low
ResNet50V2 [24]	25.6 Million	0.760	Low
InceptionV3 [58]	23.8 Million	0.779	Medium
InceptionResNetV2 [58]	55.8 Million	0.803	Medium
MobileNetV2 ($\alpha = 1.0$)[47]	3.4 Million	0.713	Low

Table 4.2: Comparison of backbone architectures when applied to the ImageNet dataset. The assessment of the complexity may be subjective.

The benchmarks in table 4.2 are taken from the reference implementations provided in the Keras-API of TensorFlow[1] and the exact number of parameters and accuracy-score my vary in other implementation.

Both of the Inception architectures yield considerably high accuracy results, but also have relatively advanced features. In contrast to other architectures, they do not follow the depth-wise block-stacking structure often used and instead consist of carefully crafted sections. They may be untypical and therefore not ideal for adaption.

As the VGG16 architecture is both, large and relatively ineffective it will be disregarded. The ResNet50V2 architecture has a fairly low number of parameters and a high accuracy score. As an additional benefit, the ResNet architecture is extensively researched and referenced and is often found as a baseline in research. Therefor, this architecture will be chosen as a backbone.

Should the ResNet architecture turn out to be too costly for the resources available, the MobileNet can be used as a fallback backbone architecture. The MobileNet architecture is even smaller than the ResNet architecture, but still has a reasonably good accuracy-score.

1 https://keras.io/api/applications/

4.5 IMPLEMENTATION AND BENCHMARKS

To implement the software, the TensorFlow[2] framework will be used. TensorFlow's Keras-API[3] allows constructing artificial neural networks in a convenient and adaptable fashion. As a high level API, this also allows quick experimentation and implementation of various models. Furthermore, the API-design is open and modifiable, which further helps the implementation. This is ideal, as a variety of modules need to be inserted into the architectures. Also, TensorFlow provides a number of reference implementations which can readily be used.

Concerning the benchmarks, several datasets will be used. To test the image classification prototypes, the CIFAR-100 and the Imagenette datasets will be used. The final video classification models will be trained on HMDB51 and UCF101.

2 https://www.tensorflow.org/
3 https://keras.io/

5

IMPLEMENTATION

This chapter intends to give the reader an overview of the hardware and software used in the project and highlight the individual decisions taken during implementation. It begins by describing hardware and software used. Afterwards, a broad overview of the project is given, before the implementation details of the backbone models are discussed. Here, a first quick evaluation and comparison of the models will be done.

After the implementation of the backbone architectures, the implementation of the video classification architecture, namely the two-stream model, is discussed.

The code for this implementation can be found online at GitHub[1].

5.1 HARDWARE SPECIFICATIONS

All benchmarks are conducted in a Google Cloud Platform Compute Engine using the N1-highmen-4 profile with 4 virtual CPU-Cores[2] and 26 GB of RAM. Furthermore, an NVIDIA Tesla T4 graphics card is used.

5.2 SOFTWARE OVERVIEW

Python 3.8 and TensorFlow 2.3 were used to implement all models. The conda package manager was to organize the environments, software versions and libraries used.

The TensorFlow's Keras-API provides prebuilt classes and functions to implement artificial neural networks. The API's various layer-classes were extensively used to implement the neural networks and attention modules. Furthermore, the interface also allowed to implement attention modules as Keras-layers and use them with the existing ones.

Keras also provides various utilities. In some cases, these utilities and the package tensorflow-datasets were used to access and manage the various datasets used. Additionally, Jupyter Notebooks were used to visualize the training and display the results in a practically and graphically. This part was an optional addition, as the training can also be done from command line. Software tests were implemented with the testing framework pytest.

1 https://github.com/zr123/VideoClassificationWithAttention/tree/v1.0

2 More information about the used hardware can be found in this data sheet: https://cloud.google.com/compute/docs/cpu-platforms. As of November 2021 the N1 profiles primarily use processors of the Intel Xeon brand.

The Python interface of OpenCV[3] was used to implement various image manipulation functions. This was especially helpful for the calculation of the optical flow information.

A full listing of the packages used can be found in the `cpu-environment.yml` and `gpu-environment.yml` files in the root directory of the project.

5.3 PROJECT OVERVIEW

The final project structure is shown below. Some files have been omitted for the sake of clarity.

```
/
├── tests
│   └── ...
├── TF_modification
│   └── ...
├── VCWA
│   ├── AttentionModels.py
│   ├── Common.py
│   ├── VideoDataGenerator.py
│   ├── VideoModels.py
│   └── Attention
│       ├── AttentionGate.py
│       ├── CBAM.py
│       ├── L2PAModule.py
│       └── ResidualAttentionModule.py
├── AttentionPrototypes.ipynb
├── Preprocessing.ipynb
└── VideoModels.ipynb
```

The test and build infrastructure was implemented with GitHub actions. Having tests and automatic builds helped to uphold the project quality and detect possible problems as early as possible.

The modifications of the TensorFlow models were kept separately in the package `TF_Modifications` to separate the code licensed by the TensorFlow authors from the remaining project.

For the most part, the project is contained in the package `VCWA`. The individual attention modules are implemented in the sub-package `Attention`. The Jupyter Notebook files contained in the root structure provide functionality to easily start and monitor training and preprocessing.

5.4 BACKBONE PROTOTYPES

The majority of the work of implementation concerned the backbone architectures and the attention modules. Initially, the ResNet50v2 architecture was used as a base model.

3 https://opencv.org/

Conveniently, Keras provides an implementation which can be used and adapted[4].

The Learn to Pay Attention (L2PA) modules were implemented as a Keras-layer by extending the provided layer-parent class. Jetley et al. discuss two compatibility functions, the simple dot-product and a weighted-dot-product. Both variants were implemented and can be chosen as a hyperparameter when instantiating the module. The weighted compatibility function is used as the default value, as the authors report better results with this function.

Even though the L2PA module itself is not very complicated, it can still be difficult to use. Typically, it requires to 1) construct a 2D-CNN as one normally would; 2) create the attention modules and connect them with inputs from various stages of the network, and 3) concatenate the outputs and finalize the new model. This structure differs strongly from the usual block-wise construction of neural networks and may be unintuitive for readers unfamiliar with the module.

The L2PA-implementation diverges slightly, as a densely connected mapping layer was consistently used to scale the global features vector to the size of the local feature vector. This layer is not strictly necessary, when both feature vectors are already of the same size. The adjustment was made to have a similar amount of complexity in each module. Finally, problems with over-fitting were found during testing and to combat this problem a dropout of 0.5 as added to the final dense fusion layer.

The attention gated module was also implemented as a child of the Keras-layer-class. This module was designed with hyperparameters for the internal channels (c_{int}), the attention function and a boolean parameter to enable grid-attention. The implementation was uncomplicated, as the compatibility function could be constructed out of the `Conv1D`-, `Conv2D`- and `Dense`-layer-classes provided by Keras. The grid-attention was implemented by nearest neighbor upsampling the data of the global features.

As attention functions, the sigmoid, softmax and the pseudo softmax functions proposed by the authors were implemented. Pseudo softmax is used as default and for all implementations. This and all other implementation details follow the original paper. The individual attention gates were combined by average-fusion.

Unlike the previous modules, the residual attention module was not implemented as a layer, but functionally as a stack of layers. This function has a larger amount of parameters, $\boldsymbol{p}$ for the number of padding-blocks before and after the module, $\boldsymbol{t}$ for the number of blocks in the trunk-branch and $\boldsymbol{r}$ for the number of blocks in the mask-branch and a parameter to change the attention function. Additionally, this function takes a parameter for the number of shortcuts and a takes a functional argument to build the residual blocks within the module. With the help of a block construction function, the residual attention module can be adjusted for a usage with various models. This block construction function constructs a residual blocks suitable for the respective model and is used to create the internal blocks within the residual attention module.

4 See https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet_v2/ResNet50V2

The entire process follows the specifications of the respective paper, including the default parameters of $p = 1, t = 2, r = 1$.

The convolutional block attention module (CBAM) was implemented like the residual attention module as a function. Just as before, this module could be build primarily from components of the Keras-API. A few operations were directly implemented using TensorFlow's tensor functions. Otherwise, the reduction ratio r of the channel attention block and the kernel size k of the spatial attention block were implemented as hyperparameters. As default parameters, the recommended values of $r = 16$ and $k = 7$ were adopted.

The described attention modules were used to create six backbone networks. Table 5.1 shows the names of the models and compares their complexity.

Model	Parameters	FLOPs
ResNetV2	23,769,700	6,988,783,832
ResNetV2 + L2PA (Compatibility: weighted, Fusion: Dense 1024)	28,395,620	6,999,088,113
ResNetV2 + gated attention (c_{int}=32, Fusion: Dense 1024)	28,581,136	7,052,632,493
ResNetV2 + gated grid attention (c_{int}=32, Fusion: Dense 1024)	28,581,136	7,090,390,451
ResNetV2 + residual attention (p=0, t=2, r=1)	33,274,980	12,879,604,452
ResNetV2 + CBAM (r=16.0, k=7)	23,943,933	7,000,117,000

Table 5.1: Comparison of ResNetV2 backbone architecture implementations for 100 classes

In each version of the model, attention modules were inserted in three different places. The position of each attention module was chosen in according to recommendations by their respective authors.

As can be seen in table 5.1, most modules significantly increase the number of parameters of the model. The only exception is the CBAM, which increases the total number of parameters only by a relatively small amount. Concerning the computational cost, the residual attention module is the only module which significantly increases the number of floating-point operations for a singular forward-propagation. This is despite the padding hyperparameter p already being reduced down to 0.

In preliminary tests, ResNetV2 turned out to be still fairly time-consuming and resource-hungry. Consequently, the MobileNetV2 backbone was also implemented. See table 5.2. Using the MobileNetV2 architecture reduces the number of parameters and floating point operations roughly by a tenth. Interestingly, not all models are affected by this equally. Especially, the L2PA module is surprisingly memory-intensive in this context.

Model	Parameters	FLOPs
MobileNetV2 (α=1.0)	2,386,084	612,982,808
MobileNetV2 + L2PA (Compatibility: weighted, Fusion: Dense 1024)	3,784,420	615,937,969
MobileNetV2 + gated attention (c_{int}=32, Fusion: Dense 1024)	3,126,928	620,158,221
MobileNetV2 + gated grid attention (c_{int}=32, Fusion: Dense 1024)	3,126,928	631,959,315
MobileNetV2 + residual attention (p=0, t=0, r=1)	2,629,044	687,043,220
MobileNetV2 + CBAM (r=16.0, k=7)	2,387,870	614,629,354

Table 5.2: Comparison of MobileNet backbone architecture implementations for 100 classes

This is because this module strongly scales with the number of channels, which stay relatively high in the MovileNetV2 model.

As previously discussed, the size of the residual attention module was further reduced to bring the model down to a similar size of the others. To compensate, the module was implemented parallel to the existing convolutional blocks in the network. But, the module still has the largest increase in floating point operations.

Also noteworthy, the CBAM is very lightweight both in memory and computation cost, again.

The image augmentation was implemented using utility functions provided by TensorFlow. Random rotation of up to 20 degrees, a random shear rotation of up to 20 degrees, a random zoom range of up to 20% and random horizontal flips were used.

As backpropagation algorithm, the default SGD with a learning rate of 0.1, a momentum of 0.9 and weight decay of 0.0001 was used. The batch size is 64. Initially, the ADAM was used, but this lead to premature saturation problems.

5.5 VIDEO CLASSIFICATION MODELS

The two-stream model was uncomplicated to implement. Keras provides a `TimeDistributed`-layer class, which facilitates reusing a stack of layers or an embedded model repeatedly for temporal data. This is exactly the behavior wanted for two-stream models, where a backbone model is applied to a time-distributed set of frames. The same functionality can be used to implement the optical flow stream.

To be able to use different backbones and model versions, the two-stream models are assembled using a function, which takes a spatial stream model and a temporal stream model as arguments. This allows for maximum flexibility, as the backbone networks can also be trained individually before assembling the two-stream model. Additionally, this way the backbone models can be evaluated individually at an early stage, and various

fusion techniques can be tested relatively late, potentially saving training time.

The stacked dense optical flow was calculated using the OpenCV's out-of-the-box implementation of the DualTVL1 algorithm. The usage of optical flow comes with two challenges. Firstly, calculating the stacked optical flow is computationally expensive and can typically not be done dynamically during training. Secondly, stacking dense optical flow leads to very large datasets. For example, for a video with 25 frames and a resolution of 224x224, the 10-fold stacked optical flow tensor has the dimensions (15, 224, 224, 20). Additionally, writing this data to disk is difficult because of the unusual content of the data. Images and video-frames typically have 3-color channels with 8-bit integer data. The optical flow tensor has 20 channels with 32-bit floating point data, resulting in up to 80 times more data. Finally, the usual compression and file formats for images and videos can not be used for this data.

It is important to address these issues, as reading the data from disk can become the bottleneck during training, if the amount of data is too large. Firstly, 5-fold stacked optical flow was used. Simonyan and Zisserman showed that 10-fold optical flow still results in better accuracy scores, but the gain is comparatively small.[54] 5-fold stacked optical flow therefore constitutes a reasonable compromise between quantity of data and quality of results. After calculating the dense optical flow, the results were directly converted to 8-bit integers. This conversion is lossy, but reduces the total amount of data effectively. Finally, loss-free zip-compression was used to write the data to disk.

Another major issue was the implementing the training itself. For the training of the image models, the datasets were small enough to be stored in memory. This is not always feasible for the much larger video data. Consequentially, a class VideoDataGenerator was written to generate batches for the training. This class was designed to work with video data and optical flow data individually and simultaneously, so the spatial stream model, optical stream model and two-stream model can all be trained using this class. For this task, the class was also designed to be compatible with various data formats.

Finally, image augmentation support was also implemented.

For the training, the video datasets were down-scaled to 25 frames, which were uniformly sampled from each video. As before, the frames were augmented by random rotation of up to 20 degrees, random shear rotation of up to 20 degrees, a random zoom of up to 20 % and finally random horizontal flips of the frame. Also, just as before, the SGD backpropagation algorithm with a learning rate of 0.1, a momentum of 0.9 and weight decay of 0.0001 was used.

5.6 ATTENTION VISUALIZATION

Extracting and visualizing the attention calculated within the modules also was an important part of the implementation. To a certain extent, this part of the implementation was also the most straight-forward and accompanied by the fewest design decisions.

Three of the attention modules directly calculate an attention heat-map as part of their functionality. Here it is only a technical matter of accessing this information during validation.

For the CBAM, the module does not internally use a visual attention mechanism. Instead, Grad-CAM has to be used to calculate a heat map. This was function was implemented accordingly. Beneficially, Grad-CAM can also be used to visualize the behavior of the unmodified MobileNetV2.

Since each module is inserted at three different stages of the network, three attention heat maps are created for each input. Usually these heat maps differ in size with each other and from the input. To better visualize the attention, some additional utility was implemented, to combine and compare this information.

6

FINDINGS

This chapter summarizes the results of this thesis. Firstly, this concerns the observations made when training the prototype image classification architectures. Afterwards, the visual attention heat maps produced by the modules are compared. Finally, the topic of benchmarks is addressed.

6.1 TRAINING

The image prototype architectures were evaluated on the CIFAR-100 and Imagenette datasets. Both datasets have the advantage of being relatively small and having short training times. No pretraining was performed.

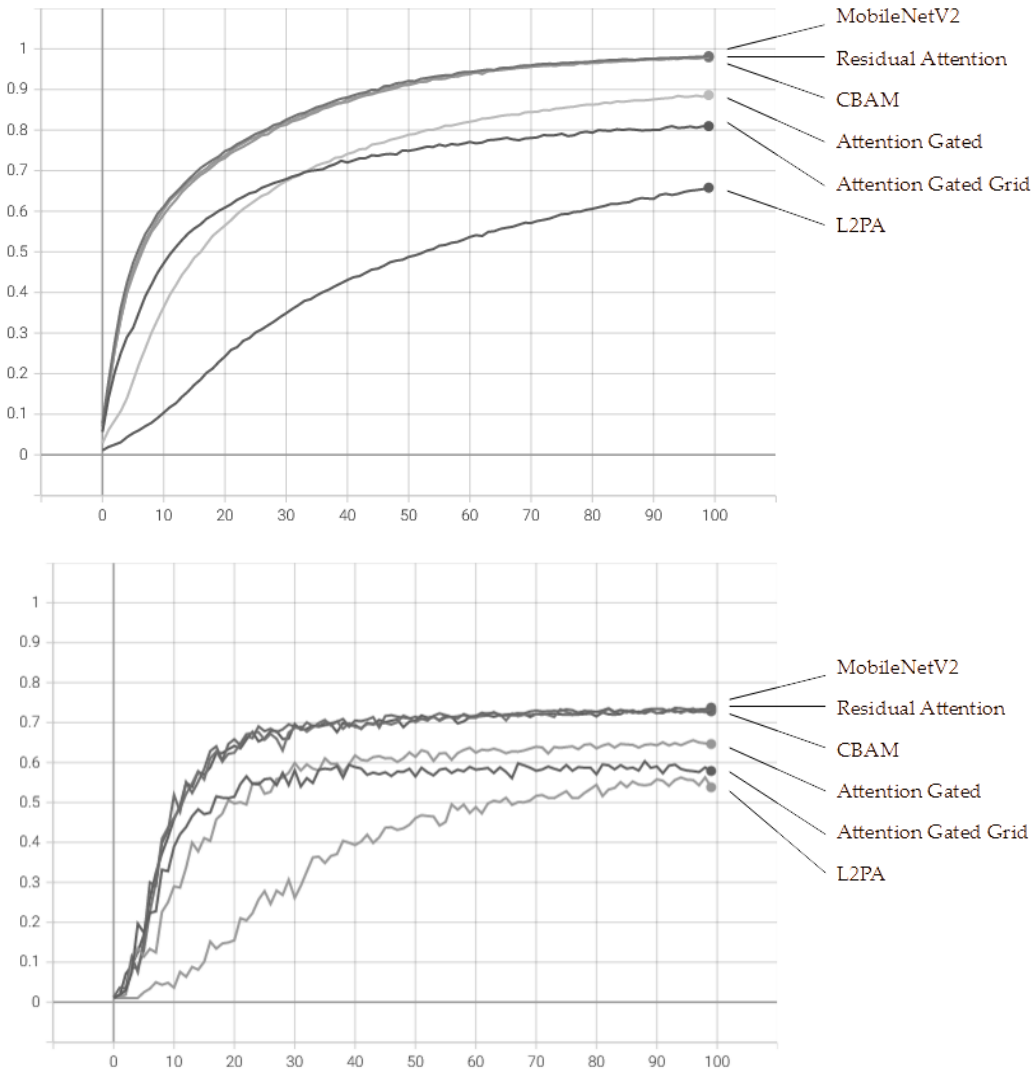


Figure 6.1: Training curve (top) and test curve (bottom) for the CIFAR-100 dataset.

The first observation to be made is the L2PA module, the attention gated module and the attention gated grid module have a negative impact on the overall performance of the network. These three adaptions of the MobileNetV2 architecture perform worse than the unmodified original network.

The residual attention module and the CBAM in comparison bring slight improvements to the base network. The following table better illustrates the performance gains. See table 6.1.

Model	CIFAR-100 Top-1	CIFAR-100 Top-5	Imagenette Top-1	Imagenette Top-5
MobileNetV2	0.7285	0.9254	0.8158	0.9781
MobileNetV2 + L2PA	0.5379	0.8108	0.7804	0.9628
MobileNetV2 + Gated Attention	0.6463	0.8179	0.7659	0.9483
MobileNetV2 + Gated Grid Attention	0.5788	0.8377	0.7203	0.906
MobileNetV2 + Residual Attention	0.7324	0.9306	0.8504	0.9857
MobileNetV2 + CBAM	0.7376	0.9299	0.8262	0.9824

Table 6.1: Comparison of test scores of the backbone prototypes.

As can be seen, the models augmented with residual attention modules and with CBAM outperform the basic MobileNetV2 for both datasets.

6.2 ATTENTION

First, the attention for the CIFAR-100 is evaluated. Table 6.2 shows the results, for each of the attention modules. The stage refers to the position of the attention module within the network. The table also shows a combined attention map overlay, constructed by up-sampling and averaging the attention results, and overlaying the original image. For the unmodified MobileNetV2 and the network augmented with CBAM, Grad-CAM is used to visualize create the heat maps.

An up-scaled version of the original image can be found in appendix A.1.

A number of observations can be made. Firstly, it is apparent the L2PA module did not work as intended for this dataset and architecture. Instead of developing a visual attention map, it seemingly only concentrated on singular spatial positions.

The gated attention module and gated grid attention module also developed this issue for the late state module, but otherwise produced attention maps. Interestingly, the heat map produced by the grid attention module exhibits borders around the lines of the grid.

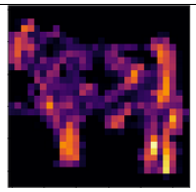
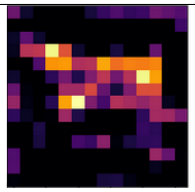
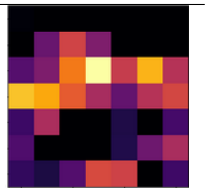
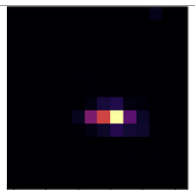
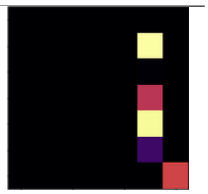
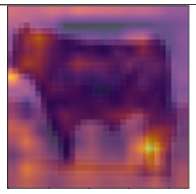
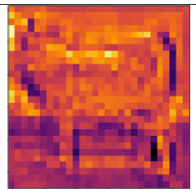
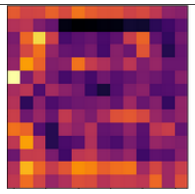
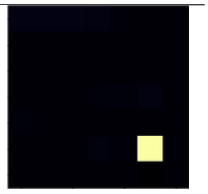
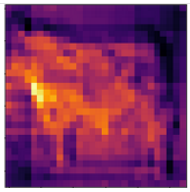
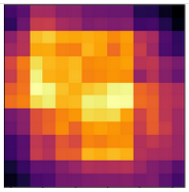
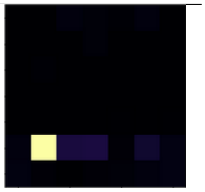
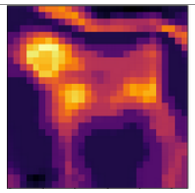
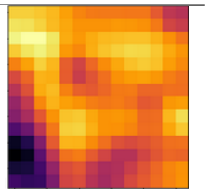
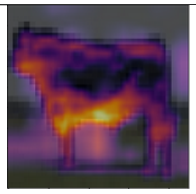
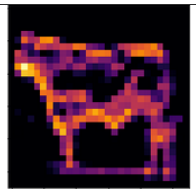
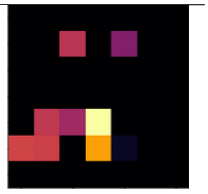
Model	Attention			
	combined	early stage	medium stage	late stage
MobileNetV2 (Grad-CAM)				
MobileNetV2 + L2PA				
MobileNetV2 + Gated Attention				
MobileNetV2 + Gated Grid Attention				
MobileNetV2 + Residual Attention				
MobileNetV2 + CBAM (Grad-CAM)				

Table 6.2: Attention maps extracted from the models for the CIFAR-100 dataset.

The residual attention module produced attention, but it is open for discussion if it worked as intended. Especially the early stage module highlights the background, rather than the object of interest. This is the exact opposite of the desired behavior. Also, the late stage module produced attention which is difficult to interpret, as it is fairly evenly distributed.

Finally, the class activation heat map produced by Grad-CAM for the unmodified model and the CBAM version of the model are somewhat similar. Both clearly outline the object of interest at an early stage, but continue to lose focus when extracted from later stages of the network.

The attention results are additionally analyzed on the Imagenette dataset. This dataset has a much lower count of classes, but a higher resolution, and thus allows different insights into the behavior of the modules. See table 6.3.

A higher resolution version of the original image can be found in appendix A.2.

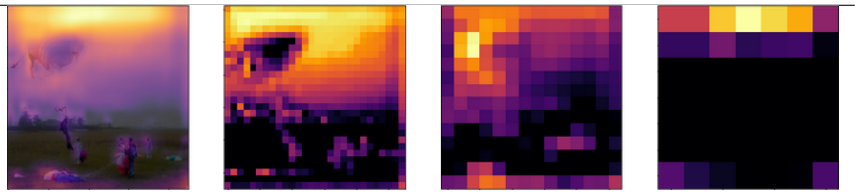
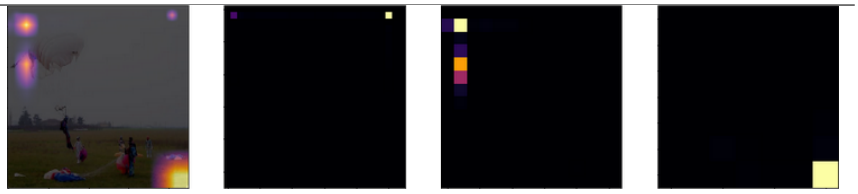
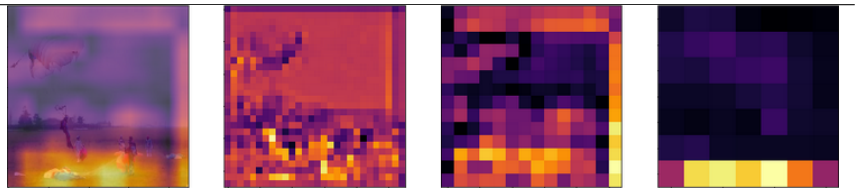
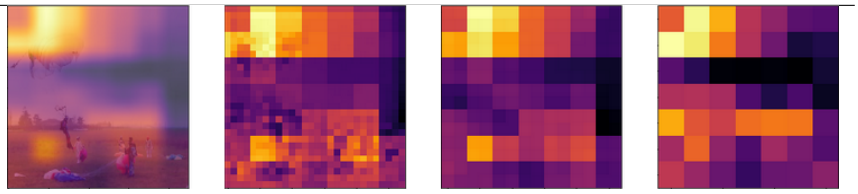
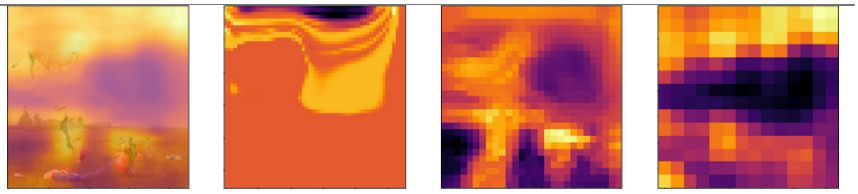
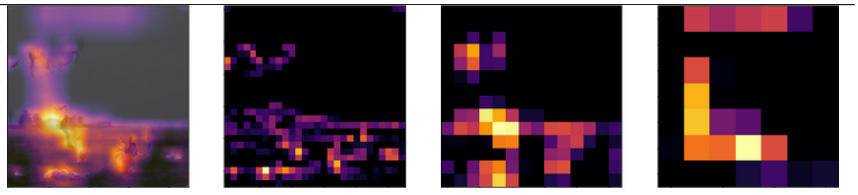
Model	Attention			
	combined	early stage	medium stage	late stage
MobileNetV2 (Grad-CAM)				
MobileNetV2 + L2PA				
MobileNetV2 + Gated Attention				
MobileNetV2 + Gated Grid Attention				
MobileNetV2 + Residual Attention				
MobileNetV2 + CBAM (Grad-CAM)				

Table 6.3: Attention maps extracted from the models for the Imagenette dataset.

When trained on Imagenette the L2PA module also did not develop the intended behavior. In contrast, the gated attention modules, especially the gated grid attention, did not exhibit the same issues at the late stage. A possible explanation is the difference in resolution between CIFAR-100 and Imagenette.

The residual attention modules show the same problems they did for CIFAR-100. The attention maps seem highly general and unspecific. Especially the early stage module seems to have produce no valuable attention.

Finally, the CBAM module seems to work as intended. The Grad-CAM for the unmodified MobileNetV2 focuses strongly on background information, in this case the sky. The model enhanced with CBAM instead focuses the objects of interest in the sky and on the ground.

6.3 BENCHMARKS

Regrettably, due to hardware constrains, the original goal of producing benchmarks could not be reached. Initially, during the conception phase, the required resources were underestimated. Even though the computational costs were scaled back, by using a small backbone architecture, relatively small video datasets

A major limiting factor was the internal video RAM of the graphics cards used. When training models, the video memory needs to be large enough to accommodate the tensors operated on. If these tensors are too large, the solution is to reduce the tensor size by choosing a smaller model, reducing the amount of data or by reducing the batch size. Despite scaling down the architecture and the amount of data, this did not prove to be enough. The last approach of reducing the batch size, technically solved this issue, but lead to further problems. Small batch sizes lead to an inaccurate gradient and unstable training. If the batch size is too small, the computed gradient will be too inaccurate and the training will fail. This was the issue in this case.

7

DISCUSSION

This chapter summarizes and comments on the results and observations of the previous chapter. First, the attention modules addressed individually, to cover their distinct advantages and disadvantages. Afterwards, some general challenges of attention are discussed. This section also investigates problems specific to video classification and other issues that may arise but were not encountered in this thesis. Finally, the relationship between visual attention modules and post-hoc attention like GradCAM needs to be reviewed.

7.1 ATTENTION MODULES

This section discusses the implemented attention modules individually and addresses their individual strengths and problems.

7.1.1 *Learn to pay Attention*

When evaluating the theory behind L2PA modules, a few important observations and criticisms need to be made.

First, the authors tested this attention module on the by 2018 outdated VGG16 architecture[55]. At the time, more efficient and better performing architectures like ResNet[24] were readily available.

Secondly, Jetley et al. primarily use the CIFAR-10/CIFAR-100 dataset[37] and a downscaled version of the CUB-200-2011 dataset[65]. With 32x32 and 80x80 pixels, both datasets used have a small resolution in comparison to other common datasets available in 2018. For example, the VGG16 architecture was originally proposed with a 224x224 pixel version of the ImageNet dataset in mind[55]. It is fully clear why these datasets were chosen.

Finally, the attention modules introduce an information bottleneck into the network in the form of the weight vector $\mathbf{u}$. This may negatively impact the abstraction ability of the network. If and how this problem practically impairs a network needs to be verified in further empirical testing with different base architectures and datasets.

Practically, the L2PA did not produce visually interpretable attention and negatively impacted the performance when integrated into the MobileNetV2 architecture. While the VGG16 and MobileNetV2 architectures differ in a multitude of points, the most likely cause of this misbehavior are the residual connections. However, this assumption is speculative and more research is necessary to be unequivocally sure of the exact causes.

7.1.2 *Attention-Gated Network*

When examining the gated attention mechanism, largely the same observations can be made, which were made for the L2PA-module. First and foremost, the bottleneck-issue is addressed by increasing the number of trainable parameters and by introducing a scaling mechanism with, C_{int}. However, Schlemper et al. primarily base their research on classification networks on the fairly small SonoNet[3] architecture with SonoNet-8, SonoNet-16 and SonoNet-32, which have only 0.16M, 0.65M and 2.58 parameters respectively.[48] With 2694 images and a final resolution of 208x272 fetal ultrasounds, the dataset they used is also fairly small and highly specific. Ideally, the authors should have provided research into larger datasets like ImageNet and more modern network architectures like ResNets.

This thesis conducted this research in a limited fashion, by applying the mechanism to MobileNetV2 and with the CIFAR-100 and Imagenette datasets. While it may be premature to dismiss this attention mechanism entirely based on these results, the preliminary findings are clearly unfavorable. The resulting network is slower to train and performs worse than its unmodified original.

7.1.3 *Residual Attention Network*

The authors of the residual attention network verified their attention mechanism on a variety of modern image classification architectures like ResNet, ResNeXt and Inception-ResNet, and on a variety of datasets like CIFAR-100 and ImageNet.[66] This can be seen as strong evidence for the functionality and maturity of this module.

As a major point of criticism, it has to be pointed out the residual attention module comes with a high price in terms of network size and computational cost. However, each module can also flexibly be scaled up or down with the provided hyperparameters to fit specific needs.

Another downside of this module is the generated attention itself, which seems to be inconsistent and highly dependent on the placement within the model. However, this issue likely can be solved during modelling by placing the module in appropriate positions.

7.1.4 *Convolutional Block Attention Module*

Woo et al. provide strong evidence for their convolutional block attention module, having tested it with the ResNet, WideResNet, ResNeXt, MobileNet and VGG architectures and the ImageNet dataset.[69]

The main advantage of this module is how lightweight it is, coming with a very small footprint in terms of memory and computations. At the same time, it provides

a considerable benefit during training, helping the network focus and increasing the overall accuracy.

The main disadvantage is CBAM does not provide visual attention on its own, and additional techniques like Grad-CAM have to be used to calculate an attention heat map.

7.2 CHALLENGES

Adopting visual attention for architectures like regular two-stream networks is relatively easy, as the backbone architecture can be directly modified. When using other architectures like (2+1)D networks, on the other hand, one may encounter difficulties, as the attention mechanisms may not necessarily translate into 3-dimnesional convolutions. This is a problem, as techniques like inflation translate 2-dimensional convolutions into 3-dimensional convolutions, also posing a problem which needs to be addressed when using attention. So even architectures which are usually well suited for visual attention, like two-stream models, may be less suitable when this technique is used.

7.3 RELATION TO POST-HOC ATTENTION

Finally, when investigating visual attention, techniques like Grad-CAM pose a fundamental question. Are learnable visual attention even necessary?

Creating visual heat maps from 2-dimensional convolutions with Grad-CAM can be done with any 2D-CNN. This technique can also flexibly be used on any layer within the network. Lastly, this technique is very fast and can be used to create heat maps for a multitude of inputs reasonably quickly.

Visual attention mechanisms on the other hand require the network to be retrained to provide usable information. In the worst case, the only information the attention module provides is about the module itself being misplaced, requiring repeated retraining with trail-and-error. To some extent, it seems, visual attention is self-serving, rather than helping the goal of model diagnosis.

In comparison, forgoing visual attention in favor of non-visual attention mechanisms like CBAM seems to be more flexible, more lightweight and less prone to errors. Furthermore, the behavior can directly be compared to the original network. In this regard, learnable visual attention seems to have few advantages.

8

CONCLUSION AND OUTLOOK

This chapter finalizes this thesis and formulates what conclusions can and can not be drawn from the results. Also, this thesis highlights a few research topics worth investigating. In the outlook section, these research topics are addressed and summarized.

8.1 CONCLUSION

The results of this research into visual attention has been, to some extent, underwhelming. While technically feasible, visual attention modules also come with a number of caveats. Even the residual attention module, which yielded the best results, comes with a significant increase in memory-consumption and computations, requires careful placement within the network and may not always produce easily interpretable visual attention.

Yet, research also clearly shows attention generally is a useful tool for artificial neural networks. Especially in natural language processing, attention is now a fundamental part of modern architectures. Components like the convolutional block attention module show attention can successfully be applied in computer vision and improve results at relatively little costs.

8.2 OUTLOOK

In conclusion, as a way forward for attention in the field of neural computer vision, it might be the best course of action to forego visual attention and focus research on more lightweight attention components. Also, integrating attention as a more fundamental building block, rather than a late addition, as it is done in the transformer family of architectures, is another interesting path for attention.

Finally, there is still much research to be done, both concerning the application of attention in the field of computer vision, and concerning human action recognition in general.

This thesis focused on the application of spatial attention. Another interesting topic is the application of temporal attention mechanisms with video data.

Part II

APPENDIX

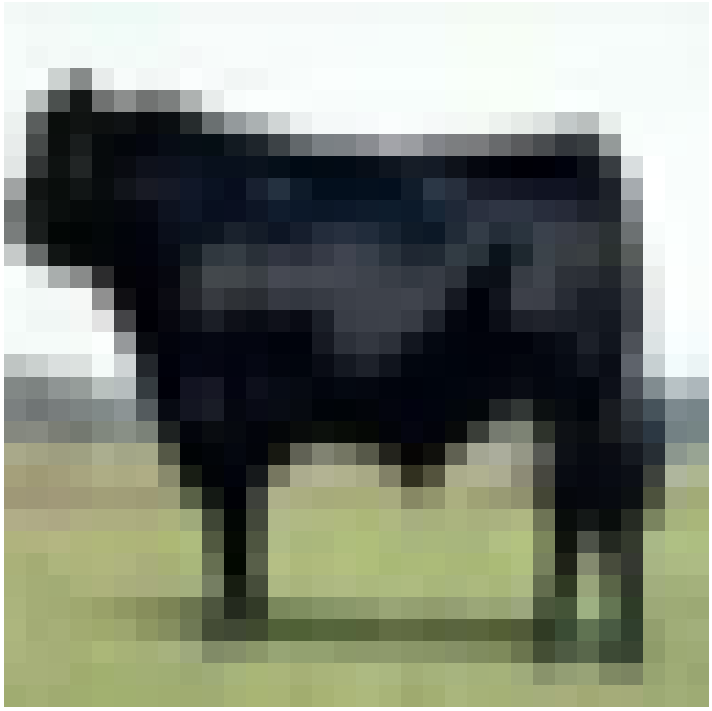


Figure A.1: A CIFAR-100 image of the class *cattle*

Figure A.2: A Imagenette image of the class *parachute*

BIBLIOGRAPHY

[1] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. "Youtube-8m: A large-scale video classification benchmark." In: *arXiv preprint arXiv:1609.08675* (2016).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." In: *arXiv preprint arXiv:1409.0473* (2014).

[3] Christian F Baumgartner, Konstantinos Kamnitsas, Jacqueline Matthew, Tara P Fletcher, Sandra Smith, Lisa M Koch, Bernhard Kainz, and Daniel Rueckert. "SonoNet: real-time detection and localisation of fetal standard scan planes in freehand ultrasound." In: *IEEE transactions on medical imaging* 36.11 (2017), pp. 2204–2215.

[4] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. "High accuracy optical flow estimation based on a theory for warping." In: *European conference on computer vision*. Springer. 2004, pp. 25–36.

[5] Joao Carreira and Andrew Zisserman. "Quo vadis, action recognition? a new model and the kinetics dataset." In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[6] Jose M Chaquet, Enrique J Carmona, and Antonio Fernández-Caballero. "A survey of video datasets for human action and activity recognition." In: *Computer Vision and Image Understanding* 117.6 (2013), pp. 633–659.

[7] Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. "An attentive survey of attention models." In: *arXiv preprint arXiv:1904.02874v2* (2020).

[8] Xinlei Chen and Abhinav Gupta. "Webly supervised learning of convolutional networks." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1431–1439.

[9] Dan Claudiu Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. "Flexible, high performance convolutional neural networks for image classification." In: *Twenty-second international joint conference on artificial intelligence*. 2011.

[10] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. "CoAtNet: Marrying Convolution and Attention for All Data Sizes." In: *arXiv preprint arXiv:2106.04803* (2021).

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[12] Li Deng. "The mnist database of handwritten digit images for machine learning research." In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." In: *arXiv preprint arXiv:1810.04805* (2018).

[14] Santosh K Divvala, Ali Farhadi, and Carlos Guestrin. "Learning everything about anything: Webly-supervised visual concept learning." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 3270–3277.

[15] Haodong Duan, Yue Zhao, Yuanjun Xiong, Wentao Liu, and Dahua Lin. "Omni-sourced webly-supervised learning for video recognition." In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16*. Springer. 2020, pp. 670–688.

[16] David Fleet and Yair Weiss. "Optical flow estimation." In: *Handbook of mathematical models in computer vision*. Springer, 2006, pp. 237–257.

[17] Andrea Galassi, Marco Lippi, and Paolo Torroni. "Attention in natural language processing." In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[18] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM." In: *Neural computation* 12.10 (2000), pp. 2451–2471.

[19] Shreyank N Gowda, Marcus Rohrbach, and Laura Sevilla-Lara. "SMART Frame Selection for Action Recognition." In: *arXiv preprint arXiv:2012.10671* (2020).

[20] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. "The" something something" video database for learning and evaluating visual common sense." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5842–5850.

[21] Ben Graham. "Sparse 3D convolutional neural networks." In: *arXiv preprint arXiv:1505.02890* (2015).

[22] Kaiming He, Ross Girshick, and Piotr Dollár. "Rethinking imagenet pre-training." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4918–4927.

[23] Kaiming He and Jian Sun. "Convolutional neural networks at constrained time cost." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 5353–5360.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[25] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[26] Berthold KP Horn and Brian G Schunck. "Determining optical flow." In: *Artificial intelligence* 17.1-3 (1981), pp. 185–203.

[27] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." In: *arXiv preprint arXiv:1704.04861* (2017).

[28] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-excitation networks." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7132–7141.

[29] Saumya Jetley, Nicholas A Lord, Namhoon Lee, and Philip HS Torr. "Learn to pay attention." In: *arXiv preprint arXiv:1804.02391* (2018).

[30] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J Black. "Towards understanding action recognition." In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 3192–3199.

[31] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. "3D convolutional neural networks for human action recognition." In: *IEEE transactions on pattern analysis and machine intelligence* 35.1 (2012), pp. 221–231.

[32] M Esat Kalfaoglu, Sinan Kalkan, and A Aydin Alatan. "Late temporal modeling in 3d cnn architectures with bert for action recognition." In: *European Conference on Computer Vision*. Springer. 2020, pp. 731–747.

[33] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. "Large-scale video classification with convolutional neural networks." In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014, pp. 1725–1732.

[34] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. "Transformers in Vision: A Survey." In: *arXiv preprint arXiv:2101.01169* (2021).

[35] Yu Kong and Yun Fu. "Human action recognition and prediction: A survey." In: *arXiv preprint arXiv:1806.11230* (2018).

[36] Yu Kong, Zhiqiang Tao, and Yun Fu. "Deep sequential context networks for action prediction." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1473–1481.

[37] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images." In: (2009).

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[39] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. "HMDB: a large video database for human motion recognition." In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2011.

[40] John Boaz Lee, Ryan A Rossi, Sungchul Kim, Nesreen K Ahmed, and Eunyee Koh. "Attention models in graphs: A survey." In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 13.6 (2019), pp. 1–25.

[41] Bangli Liu, Haibin Cai, Zhaojie Ju, and Honghai Liu. "RGB-D sensing based human action and interaction analysis: A survey." In: *Pattern Recognition* 94 (2019), pp. 1–12.

[42] Elizbar A Nadaraya. "On estimating regression." In: *Theory of Probability & Its Applications* 9.1 (1964), pp. 141–142.

[43] Keiron O'Shea and Ryan Nash. "An introduction to convolutional neural networks." In: *arXiv preprint arXiv:1511.08458* (2015).

[44] Javier Sánchez Pérez, Enric Meinhardt-Llopis, and Gabriele Facciolo. "TV-L1 optical flow estimation." In: *Image Processing On Line* 2013 (2013), pp. 137–150.

[45] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[46] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. "Imagenet large scale visual recognition challenge." In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[47] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[48] Jo Schlemper, Ozan Oktay, Liang Chen, Jacqueline Matthew, Caroline Knight, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. "Attention-gated networks for improving ultrasound scan plane detection." In: *arXiv preprint arXiv:1804.05338* (2018).

[49] Jo Schlemper, Ozan Oktay, Michiel Schaap, Mattias Heinrich, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. "Attention gated networks: Learning to leverage salient regions in medical images." In: *Medical image analysis* 53 (2019), pp. 197–207.

[50] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. "Grad-cam: Visual explanations from deep networks via gradient-based localization." In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[51] Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. "Grad-CAM: Why did you say that?" In: *arXiv preprint arXiv:1611.07450* (2016).

[52] Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. "Action recognition using visual attention." In: *arXiv preprint arXiv:1511.04119* (2015).

[53] Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning." In: *Journal of Big Data* 6.1 (2019), pp. 1–48.

[54] Karen Simonyan and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos." In: *arXiv preprint arXiv:1406.2199* (2014).

[55] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." In: *arXiv preprint arXiv:1409.1556* (2014).

[56] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. "UCF101: A dataset of 101 human actions classes from videos in the wild." In: *arXiv preprint arXiv:1212.0402* (2012).

[57] Jonathan Stroud, David Ross, Chen Sun, Jia Deng, and Rahul Sukthankar. "D3d: Distilled 3d networks for video action recognition." In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 625–634.

[58] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. "Inception-v4, inception-resnet and the impact of residual connections on learning." In: *Thirty-first AAAI conference on artificial intelligence*. 2017.

[59] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[60] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114.

[61] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. "Learning spatiotemporal features with 3d convolutional networks." In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4489–4497.

[62] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. "A closer look at spatiotemporal convolutions for action recognition." In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018, pp. 6450–6459.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In: *arXiv preprint arXiv:1706.03762* (2017).

[64] Andreas Veit, Michael J Wilber, and Serge Belongie. "Residual networks behave like ensembles of relatively shallow networks." In: *Advances in neural information processing systems* 29 (2016), pp. 550–558.

[65] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. "The caltech-ucsd birds-200-2011 dataset." In: (2011).

[66] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. "Residual attention network for image classification." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3156–3164.

[67] Lei Wang, Du Q Huynh, and Piotr Koniusz. "A comparative review of recent kinect-based action recognition algorithms." In: *IEEE Transactions on Image Processing* 29 (2019), pp. 15–28.

[68] Geoffrey S Watson. "Smooth regression analysis." In: *Sankhyā: The Indian Journal of Statistics, Series A* (1964), pp. 359–372.

[69] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. "Cbam: Convolutional block attention module." In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 3–19.

[70] Jianxin Wu. "Introduction to convolutional neural networks." In: *National Key Lab for Novel Software Technology. Nanjing University. China* 5.23 (2017), p. 495.

[71] Lu Xia, Chia-Chih Chen, and Jake K Aggarwal. "View invariant human action recognition using histograms of 3d joints." In: *2012 IEEE computer society conference on computer vision and pattern recognition workshops*. IEEE. 2012, pp. 20–27.

[72] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. "Aggregated residual transformations for deep neural networks." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1492–1500.

[73] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. "Show, attend and tell: Neural image caption generation with visual attention." In: *International conference on machine learning*. PMLR. 2015, pp. 2048–2057.

[74] Ian T Young, Jan J Gerbrands, and Lucas J Van Vliet. "Fundamentals of image processing." In: (1998).

[75] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. "Beyond short snippets: Deep networks for video classification." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 4694–4702.

[76] Sergey Zagoruyko and Nikos Komodakis. "Wide residual networks." In: *arXiv preprint arXiv:1605.07146* (2016).

[77] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." In: *arXiv preprint arXiv:1409.2329* (2014).

[78] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. "Self-attention generative adversarial networks." In: *International conference on machine learning*. PMLR. 2019, pp. 7354–7363.

[79] Hong-Bo Zhang, Yi-Xiang Zhang, Bineng Zhong, Qing Lei, Lijie Yang, Ji-Xiang Du, and Duan-Sheng Chen. "A comprehensive survey of vision-based human action recognition methods." In: *Sensors* 19.5 (2019), p. 1005.

[80] Jing Zhang, Wanqing Li, Philip O Ogunbona, Pichao Wang, and Chang Tang. "RGB-D-based action recognition datasets: A survey." In: *Pattern Recognition* 60 (2016), pp. 86–105.

[81] Zhengyou Zhang. "Microsoft kinect sensor and its effect." In: *IEEE multimedia* 19.2 (2012), pp. 4–10.

[82] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. "Learning deep features for discriminative localization." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2921–2929.