School of Computing

FACULTY OF ENGINEERING AND
PHYSICAL SCIENCES

UNIVERSITY OF LEEDS

# Final Report

## Emotion Recognition using Deep Learning on a GPU

**Andrzej Miskow**

**Submitted in accordance with the requirements for the degree of
BSc Computer Science with Artificial Intelligence**

**2021/2022**

**COMP3931 Individual Project**

The candidate confirms that the following have been submitted*:*

| **Items** | **Format** | **Recipient(s) and Date** |
|---|---|---|
| *Final Report* | *Hard copy and PDF file* | *PDF uploaded to Minerva (27/07/22)* |
| *Link to online code repository* | *URL* *https://github.com/AndrzejMiskow/comp3931-emotion-recognition* | *Sent to supervisor and assessor (27/04/22)* |

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

Andrzej Miskow

## Summary

*Facial emotion recognition(FER) is a famous image classification problem popularised by the FER2013 competition. The problem was initially solved via handcrafted algorithms. However, over time researchers applied deep learning methods to significantly enhance the model's performance by improving their generalisation ability.*

*Since the competition, a range of new convolution neural network architectures has been introduced, with each new architecture improving on the previous. The researchers improve the networks by modifying the overall structure or by introducing new computational blocks. On the other hand, this project displayed improvements by modifying the network's activation and optimizer functions. By implementing VGGNet, ResNet, and ResNetV2, each model's performance was further improved on the task of emotion recognition by modifying their activation functions from ReLU to ELU.*

*Additionally, the project aimed to present a new concept inspired by psychology research attention. A wide range of research has shown that the application of attention to deep neural networks increases their performance by making the network focus on the more informative features. The paper introduced the concept of channel and spatial attention, displayed how they are implemented via SEN-Net, ECA-Net, and CBAM modules, and showed how integrating the modules with existing neural networks can significantly improve their performance for the task of emotion recognition.*

# Acknowledgements

*I would like to thank my project supervisor Dr. Abdulrahman Altahhan for his continued support and calming guidance throughout this project. The regular meetings ensured the project was completed to a high standard, and his continuous support and responses to my queries made me proud of the work I have submitted.*

*I would also like to thank my project assessor Brandon Bennett for providing valuable mid-project feedback and inspiring my ethical considerations research.*

# Table of Contents

# Chapter 1
# Introduction and Background Research

## 1.1 Introduction

“The accurate analysis and interpretation of the emotional content of human facial expressions is essential for the deeper understanding of human behaviour”  (Giannopoulos et al., 2018).

Emotions are a crucial part of any inter-personal human interaction. The emotions of another person directly influence our interaction with them. A study showed that during face-to-face human communication, 7% of the information is communicated by the linguistic part, such as the spoken words, 38% is communicated by paralanguage, such as the vocal part, and 55% is communicated by the facial expressions (Mehrabian, 1968). Hence, If a computer program could successfully detect a user's emotion and adapt to their emotional state, the interaction would feel natural, as if the user was interacting with another person. In addition to this, a deeper understanding of human behaviour lets us understand why certain people take specific actions.  In 2020, the market for emotion-detection technology was worth roughly £15.8bn, and its value is predicted to be more than double that by 2024, reaching £41bn (Gifford, 2020). Because of this, emotion recognition methods have already been applied to many fields in computer science, such as human-computer interface (Corive et al., 2001), animation (Aneja et al., 2016), medicine (Chu et al., 2018), (Edwards et al., 2002) and security (Saste and Jagdale, 2017). Even though emotion recognition models are already being used, they are constantly evolving, and we are starting to see more research breakthroughs that improve the classification of emotions.

The most recent breakthrough in the field of emotion recognition is the idea of using attention to improve the accuracy of the deep learning model. The methodology behind attention-based models was inspired by how humans inspect a scene at first glance. (Rensink, 2000) has found that humans retrieve parts of the scene or objects sequentially to find the relevant information. Since neural networks attempt to mimic how the human brain works to complete the desired task, various methods were developed to imitate human attention. The discovery of these attention mechanisms helped improve the accuracy of emotion recognition models and formed the inspiration for this project.

## 1.2 Aims, Objectives and Deliverables

### 1.2.1 Aims

This project aims to create and train a deep learning model that effectively recognizes human emotions from different angles. The model will use a readymade dataset of pictures of individuals expressing seven emotions – anger, disgust, fear, happy, sad, surprise, and neutral. Then the model should classify an image as expressing one of the emotions mentioned earlier. Additionally, the project will aim to utilize publicized attention modules that have been applied to other classification tasks. The project will show how they can be integrated into a convolution network and display that attention successfully improves the classification ability of the emotion recognition model.

### 1.2.2 Objectives

- Create a deep learning model that successfully classifies seven human emotions.
- Compare how different attention mechanisms affect the classification of human emotions
- Develop a model that can classify emotions from different angles or (non-perfect data (i.e., object covering parts of the face, user not starting directly into the camera) and still achieve good performance.
- Show attention improves the performance of deep learning models.
- Prove that the models can be applied to multiple datasets and still achieve a satisfactory performance for those datasets.

### 1.2.3 Deliverables

- Final Report
- GitHub repository (URL link) of source code

## 1.3 Literature Review

### 1.3.1 Related Works

Facial emotion recognition (FER) can be performed using different inputs such as the face (Aneja et al., 2017), speech (Han et al., 2014), EEG signals (Petrantonakis and Hadjileontiadis, 2010), and text (Wu et al., 2006). More commonly, the different inputs are combined to achieve higher accuracies, such as combining EEG signals with video footage (Huang et al., 2017). Among these, facial expressions are the most popular as datasets are the easiest to collect and label, because expert knowledge is not required to classify the images. However, FER is a complex task, and even as humans, we cannot fully understand how to classify emotions.

There is an ongoing debate on how emotions can be categorized in a computational system. The most common categorization is presented through a discrete model, widely popularized through the FER2013 competition. However (Li et al., 2021) argue that applying a limited number of emotion categories does not achieve the goal of emotion recognition. Hence, a range of research papers propose using a dimensional model as its ability to classify a broader range of emotions can achieve the desired effect of emotion recognition (Kim et al., 2018),(Mao et al., 2019), (Lee et al., 2020). The proposed solutions also widely differ in their implementations.

Initially, FER methods have utilised handcrafted features or shallow learning such as local binary patterns (LBP) (Shan et al., 2009), LBP on three orthogonal planes (LBP-TOP) (Zhao and Pietikäinen, 2007), non-negative matrix factorization (NMF) (Zhi et al., 2011) and sparse learning (Zhong et al., 2012). However, with the introduction of the FER2013 challenge, the top three performing models all utilised convolutional neural networks(CNNs) (Goodfellow et al., 2015), which showed that deep learning methods should be applied to FER applications. Additionally, a range of other deep learning methods was utilised, such as RNN(Majumder et al., 2019), GAN (Luo et al., 2019), and the best performing classification method being the SVM classifier (Li and Deng, 2020).

However, with the introduction of attention and its easy integration with CNN’s models, researchers have switched their focus to using CNN classification for FER applications. The main idea of using attention for FER tasks was proposed by (Minaee et al., 2019) where they have found that most of the needed information comes from a few parts of the face (Cohn and Zlochower, 1995). They implemented spatial attention with a CNN and improved the model's performance by focusing less on irrelevant parts of the image and training the model on “where” to find the needed information. Additionally,  (Li et al., 2021) have explored using channel attention which defines “what” to look for in the model by placing a higher value on more informative features. The combination of both spatial and channel attention for FER applications has achieved state-of-the-art results. However, the implementation of their attention modules is not publicized, so the work cannot be critically analysed.

### 1.3.1 Computational Models of Emotion

Computational models of emotion (CMEs) are software systems designed to evaluate emotional stimuli, induction of emotions, and generation of emotional behaviours (Osuna et al., 2020). Since emotions are complex concepts, they are often poorly understood hence many emotion theories/models have been developed. Due to the number of theories and models, there is a lack of consistency and clarity regarding what it means to  ‘model emotions’ (Hudlicka, 2008). The book “Guidelines for Designing Computational Models of Emotions “(Hudlicka, 2011) aimed to solve this by grouping most established and adopted theories into three categories: discrete, dimensional, and componential.

**Discrete models**: classify emotions into a small set of distinct or fundamental emotions. These classes are typically easy to recognize and can be described by everyday language. A widely used and popular discrete model is the Ekman emotion model (Ekman, 1992), which specifies six basic human emotions: anger, disgust, fear, happiness, sadness, and surprise.

**Dimensional perspective models**: classify emotions in terms of a small set of latent dimensions that define a space in which emotions can be located. The most common example of the dimensional model is the valance and arousal model developed by Mehrabian and Russell developed in 1974 (Bakker et al., 2014). The dimensions described are valence (how pleasant or unpleasant a feeling is) and arousal (degree of intensity or activation of the organism). These models are continuous and can describe a more complex and broader range of emotions. However, due to its complexity, it is hard for us to interpret the output of this model.

**Componential models**: follow the theory that more complex emotions are combinations of basic emotions. The best example of this model is Plutchik's Wheel of Emotions (Plutchik, 2001), which states eight basic emotions: joy, trust, fear, surprise, sadness, anticipation, anger, and disgust. Each basic emotion has a polar opposite, such as joy is the opposite of sadness. A combination of 2 basic emotions gives a more complex emotion, such as the combination of joy and trust resulting in love. These models are not commonly used in emotion classification due to the increase in the complexity of the classification process. The increase in complexity generally leads to longer processing time which is not ideal for most emotion recognition systems.

This project will focus on a discrete model of emotions, specifically the Ekman emotion model. Choice of the discrete model will simplify the deep learning training, as each image can only fall into seven distinct classes. Additionally, the discrete model is widely used in other research papers and image classification datasets such as FER 2013, making it possible to evaluate the project against other implementations. The drawback of this model is the inability to describe more complex emotions such as love or a mix of two basic emotions.

### 1.3.2 Deep Learning

Deep learning is a subset of machine learning and relies on deep architectures consisting of multi-layered neural networks with many hidden layers. The concept of a neural network (Hopfield, 1982) was formed from a biological model of the brain proposed by Nobel laureates Hubel and Wiesel in 1959, and it aims to emulate the way human brains "learn". In deep learning, each layer in the neural network uses representation learning to learn a representation of the original input. Representations at deeper layers are formed through a combination of the lower-level features from the previous layers. This forms a feature hierarchy, and with enough layers, even complex features can be extracted (Giannopoulos et al., 2018) (More in-depth explanation of representational learning is present in Appendix C.1.1). This paper will focus on the Convolutional Neural Networks(CNNs) (Yann LeCun et al., 1989), which are very effective for image detection and classification tasks.

### 1.3.3 Convolutional Networks

Convolutional Neural Networks(CNNs) (Yann LeCun et al., 1989) are neural networks that process data with a grid-like topology. For example, an image can be represented as a 2-dimensional grid of pixels. CNNs are neural networks that use a convolution operation instead of matrix multiplications (Ian Goodfellow et al., 2016) which reduces the number of parameters at each neuron and hence improves the efficiency of the neural network. The convolution operation provides serval other benefits apart from just improved efficiency, which are explained in Appendix C.1.2 and C.1.3. CNN's are formed from a set of convolution layers being executed sequentially.

### 1.3.4 Convolutional Layer

A typical convolution layer consists of 3 stages: convolutions, non-linear activation, and pooling (Lecun et al., 2015). In the first stage, serval convolution operations are computed in parallel, producing a set of linear activations. The output from the first stage is then passed into a non-linear activation function as many problems do not have a linear solution. This stage is also referred to as the detector stage. Lastly, a pooling layer is used to reduce the output filter map.

#### Non-linear Activation Functions

The main purpose of the non-linear activation functions is to saturate (introduce non-linearity) and limit the output passed to the next layer (Albawi et al., 2018). Non-linear activation functions are used as most real-world problems require non-linear solutions. If a linear activation is used, the model can be mathematically simplified to just having one layer because summing up layers is a linear process (Волянський, 2021), excluding the need for a convolutional network.

Over the years, numerous non-linear activation functions have been utilized. In the early days, the most popular functions were sigmoid and tanh. However, these functions struggle with the problem of “vanishing gradient” (Pedamonti, 2018). This problem was discovered as neural network architectures introduced many hidden layers, i.e., the architectures became deeper. These functions are contractive almost everywhere, and the gradients at large values become almost zero. In deeper architectures, this makes the updates by the stochastic gradient descent very small or almost zero, which is the “vanishing” gradient problem.

The vanishing gradient was solved by utilizing ReLU as an activation function. Initially, the function was made to improve a restricted Boltzmann Machine (Nair and Hinton, 2010). However (Glorot et al., 2011) showed that applying ReLU as an activation function in the hidden layers could improve the learning speed of the various deep neural networks. The reason for this is that the gradients of the ReLU activation follow the identity function for positive arguments and zero otherwise, meaning that large gradient values are still used, and negative values are discarded.

$$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

However, ReLU is non-negative, and therefore it has a mean activation larger than zero. As a result, neurons with a non-zero mean activation act as a bias for the next layer. If the neurons do not cancel out, further learning causes a bias shift for the next layer. The shift in bias causes variance in weights, leading to activation function being locked to negative values. The affected neuron can no longer contribute to the network learning, and its gradient remains at zero, leaving that neuron in a “dead” state. If many neurons are affected, the model's performance is reduced. Consequently, two activation functions have been proposed that tackle the problem of bias shift differently while also solving the vanishing gradient problem.

Firstly, an ELU function was proposed that allows negative gradient values resulting in the mean unit activations being closer to zero than ReLU. For this reason, ELU achieves faster learning and significantly better generalization performance than ReLU on networks with more than five layers (Clevert et al., 2016). Like ReLU, ELU follows the identify function for positive values, whereas it utilizes the exponential function if the input is negative.

$$ELU(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

Secondly, a SELU function (Klambauer et al., 2017) was proposed that solves the issue of bias-shift through self-normalization. Through this property, activations automatically converge to a zero mean and unit variance. This convergence property makes SELU ideal for networks with many layers, and it is a further improvement to the ReLU activation function.

$$\mathrm{SEL\,U}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leqslant 0 \end{cases}$$

## Pooling

The purpose of the pooling layers is to achieve spatial invariance (explained in Appendix C.1.4) by reducing the resolution of the feature maps (Scherer et al., 2010). To be more specific, a pooling operation replaces the neural network's output at a specific location with a summary statistic of the nearby outputs. The most commonly used pooling in CNN is the max pooling(Zhouand Chellappa, 1988) operation which takes the maximum output within a rectangular neighbourhood. Other popular pooling functions include average-pooling the L2 norm of a rectangular neighbourhood or a weighted average based on the distance from the central pixel.

### Depth, Stride, and Padding

In CNN's three parameters determine the output size of feature maps: depth, stride, and zero-padding.

**Depth** of the output corresponds to the number of filters in the CNN, each of which learns to look for something different in the input locally.

**Stride** defines how many units the filter will move over the input matrix. With a stride of 1, each filter will go over the entire input space. In practice, this is ideal, especially at the early layers, as this generates a large activation map which can lead to better performance (He and Sun, 2015). On the other hand, increasing the stride reduces the output size of the feature map, reducing the memory requirement, speeding up the computation time, and avoiding the problem of overfitting.

### Fully Connected Layers

Fully Connected layers(FC) are layers where all the inputs from one layer are connected to every activation unit in the next layer. These layers are typically applied after convolution layers, where the output of the convolutions is flattened before being used as an input. The issue with fully connected layers is that they can cause overfitting and require millions of parameters (Simonyan and Zisserman, 2014). Because of this, more recent CNNs decide to use a global average pooling instead of FC layers which has fewer parameters and improves the classification accuracy on the challenging ImageNet (Russakovsky et al., 2015) dataset. Despite this (Basha et al., 2020) have shown that the application of FC layers results in an improvement in image classification on specific datasets.

## 1.3.5 Optimisers

CNN models are trained by updating the parameters of all the layers in the network through backward propagation. Training is done via optimizers applying gradient descent algorithms to minimize the loss function. The loss function defines the error between the target and predicted output and is mainly computed using cross-entropy. Using this metric, the value is near zero when desired output and predicted output are the same, which is the goal of an optimizer function (Bera and Shrivastava, 2020).

Gradient descent methods are split into three classes, with each subsequent class being an improvement of the previous one. These classes are Vanilla Gradient Descent Optimizers, Momentum-Based Gradient Descent Optimizers, and Adaptive Gradient Descent Optimizers (Kandel et al., 2020).

**Vanilla Gradient Descent Optimizers**

Vanilla Gradient Descent Optimizers are the first variation of optimizers, and the most common optimizer in this class is the **Stochastic gradient descent(SGD)**. It performs a parameter update for every sample in the dataset with a static learning rate.

The problem with SGD is that each update places the same weight on each parameter. This update equation creates fluctuations in the loss function due to the high variance between different samples, which can create noise during training (Ruder, 2016). Consequently, it is difficult to tune the learning rate in SGD as the magnitudes of different parameters vary widely (Bera and Shrivastava, 2020). Additionally, SGD can be trapped in numerous suboptimal local minima, failing to minimize the loss function fully.

**Momentum-Based Gradient Descent**

Momentum-based gradient descent aims to fix the issue of being trapped in suboptimal local minima by considering the previous gradient value. The updating rule adds a fraction of the previous update, which gives the optimizer the momentum needed to break out the local minima towards a more optimal value (Sutskever et al., 2013). For this reason, SGD with momentum is still widely used in research papers.

**Adaptive Gradient Descent Optimizers**

Adaptive Gradient Descent functions aim to solve the issue of fluctuations in the loss function, which are caused by a fixed learning rate. The fluctuations are formed as the loss function requires different updates for different parameters, especially for parameters that are not utilized often, where significant updates are needed for optimization.

**AdaGrad Optimizer**

AdaGrad optimization algorithm was the first gradient descent algorithm that implemented variable learning rates. The creators changed the learning rate value by dividing it by the sum of squares of all previous gradients. AdaGrad increases the learning rate for sparser parameters and works well for sparse gradients, such as in natural language processing and image recognition applications (Duchi JDUCHI and Singer, 2011).

The main drawback of AdaGrad is that the learning rate decreases monotonically because every added term is positive. After every iteration, the list of squared gradients expands, increasing the denominator's value. Eventually, the learning rate is so small that it stops updating the weights.

**RMSProp Optimizer**

RMSProp's main goal was to provide a solution to the diminishing learning rate of AdaGrad (Hinton et al., 2014). It proposes detecting parameters that make the loss function fluctuate and penalizes the updates on those parameters. The algorithm keeps the moving average of squared gradients for every parameter and divides the gradient by the square root of the mean square. The more the parameter diverges the gradient away from the average, the lower the learning rate. Because of this, RMSProp works well on big unfiltered data sets that contain noise.

**Adam Optimizer**

Lastly, an Adam optimizer was developed that combined the benefits of AdaGrad, RMSProp, and momentum-based SGD.  Like RMSProp, it stores the average of past gradients, representing the first moment (mean). However, Adam also utilizes the past squared gradient representing the second moment (uncentered variance). Creators of Adam have stated that Adam adds bias correction and momentum to RMSProp, which helps Adam slightly outperform it at the latter stages of optimization as gradients become sparser (Kingma and Ba, 2014).

## 1.3.6 Bias and Variance

The main goal is to create a model that can successfully classify unseen data. This performance metric is often referred to as generalization performance, and the concepts of bias and variance are closely related to it. In machine learning, bias refers to the difference between the target output and the predicted output, and variance refers to the variability of a model prediction for an output (Hastie et al., 2009).

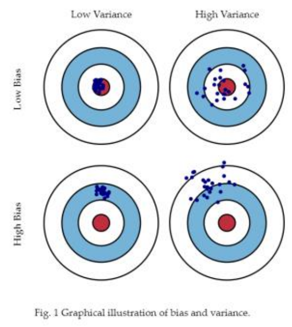


**Figure 1.1: Bias and variance representation (Scott Fortmann-Roe, 2012)**

Figure 1.1 shows a visual representation of how bias and variance affect a CNN using a bulls-eye diagram. The centre of the diagram (red region) represents a model that perfectly predicts the target value. With bias and variance, we can represent two critical problems in CNNs: overfitting or underfitting the data. When the model has low bias and high variance, the model is overfitting the data. This signifies that the model has misclassified data due to the sensitivity to small fluctuations in the training set, i.e., the CNN modelled the random noise present in the training data.

Conversely, when the model has a high bias but low variance, the model is underfitting the data. This means the model has been undertrained, and it cannot successfully classify the data. From this, we can observe that the predictive error in CNN is composed of bias, variance, and irreducible error, resulting from noise in the problem itself (Domingos, 2000). A range of improvement strategies has been researched to solve the issue of overfitting and underfitting.

### 1.3.7 Improvement Strategies

#### Data Augmentation

Data augmentation consists of modifying the original image through transformation such as rotation, translation, zooming, flipping, and cropping. This enables the model to learn augmentations that improve the ability to correctly classify unseen images (reduce overfitting) (Perez and Wang, 2017) (Mikołajczyk and Grochowski, 2018). In addition to this (Galdran et al., 2017) also explored the use of colour augmentation, such as histogram equalization, enhancing contrast or brightness, white-balancing, sharpening, and blurring, all proven to be effective at improving the performance of a model.

#### Early Stoppage

Early stoppage aims to stop training the model once the model starts overfitting the data. This concept was introduced by (Wang et al., 1995), where they have split training into three phases. At stage one, the error is dominated by approximation error where the network has hardly learned anything and is very biased. Through further training, approximation error is reduced, and complexity error is introduced, which defines the increase of variance in the model, defined as phase two. Lastly, if we continue to train the model, the error will be dominated by the complexity error known as phase three. Early stoppage aims to detect when phase two ends and the dominance of the complex error begins. If the model stops training after stage two, overfitting can be prevented. (Prechelt, 1998) explores how to find the “stoppage point” and states that it is usually done in “an ad-hoc fashion or training is stopped interactively”.

#### Batch Normalization

Batch normalization(BN) is a widely applied technique that enables faster and more stable training in deep neural networks.BN achieves this by normalizing activations in intermediate layers. The original BN paper (Ioffe and Szegedy, 2015) describes that BN reduces the internal covariate shift, which occurs when the input distribution drastically changes from one iteration to another (Shimodaira, 2000). However, (Santurkar et al., 2018) disproved this. Instead, through normalization of internal activations, BN enables larger gradient updates to occur, leading to faster convergence and helping bypass local minima. Hence, networks without BN are not resistant to significant gradient updates, resulting in diverging loss and activations growing uncontrollably with network depth. This limits the range of possible learning rates that the network can use. Hence (Bjorck et al., 2018) showed that BN also enables training at higher learning rates for faster convergence and better generalization.

### 1.3.8 Attention

#### What is Attention?

When processing a complex visual scene, human vision does not process the entire image at once. Instead, we tend to only focus on a subset of the image while ignoring the rest to speed up the visual analysis process. This process of selecting a subset of the input, and ignoring the rest, is referred to as attention (Desimone and Duncan, 1995). Additionally, (Pinto et al., 2013) have found that attention can be divided into two independent categories: bottom-up unconscious attention, referred to as saliency-based attention that operates on raw sensory input, and top-down conscious attention, which refers to the deliberate allocation of attention to certain features (Connor et al., 2004). Motivated by the research done in psychology, attention mechanisms have started to be researched in ML to improve existing models. The first implementation of attention was done in natural language processing (Bahdanau et al., 2014), where attention was applied to a recurrent network (RNN), which significantly improved its performance. Since then, attention has been widely used in a range of machine learning architectures for image caption generation (Lu et al., 2016), action recognition (Song et al., 2018), image-based analysis (Zhang et al., 2018), and machine translation (Britz et al., 2017). The first case of attention enchaining a CNN was reported in (Li et al., 2020) with application in object recognition. The main idea behind the attention mechanism in CNNs is to distinguish higher-level features from low-level features. Thus, giving higher weights to crucial features attracts the model's attention to them (Hu et al., 2018). Additionally, like in psychology, attention in ML can be split into bottom-up attention and top-down attention.

#### Bottom-up Attention

Bottom-up attention focuses on low-level features of the visual scene that stand out from its surroundings and attract attention at first glance. The low-level features are referred to as salient points. Bottom-up attention models rely on hand-crafted low-level image features to produce saliency maps (Zhao and Koch, 2013). Commonly used features include contrast (Reinagel and Zador, 1999), intensity (Krieger et al., 2000), colour (Jost et al., 2005) as well as higher-level features such as faces and text (Cerf et al., 2009).

One of the drawbacks of bottom-up attention is that it cannot automatically learn the task-specific attention features since it is purely based on low-level visual features. Because of this reason, this type of attention model is rarely utilised.

#### Top-down Attention

Top-down attention refers to attention with a predetermined purpose and requires an explicit understanding of the scene's context. Since most attention mechanisms in deep learning are designed for a specific task with known context, they all fall under the category of top-down attention.

Because attention can be applied to a wide range of applications, modifications and improvements were developed to adapt attention to specific tasks. This resulted in attention mechanisms being further categorized under four criteria that are not mutually exclusive. Each attention module can be categorised under the criteria presented in Table 1.2:

| Criterion | Type |
|---|---|
| The Softness of Attention | Soft/hard, global/local |
| Forms of Input Feature | Item-wise, location-wise |
| Input Representations | Distinctive, self, co-attention, hierarchical |
| Output Representations | Single-output, multi-head, multi-dimensional |

**Table 1.2: criteria for categorizing attention mechanism and types of attention within each criterion. (Niu et al., 2021)**

**The Softness of attention** is split into two categories *soft* and *hard* attention. The soft (deterministic) attention (Bahdanau et al., 2014) uses the weighted average of all weights to build the context vector and is differentiable hence it can be trained using back-propagation. On the other hand, hard (stochastic) attention (Xu et al., 2015) is computed stochastically from the sampled weights. It is not differentiable and has to be trained through reinforcement learning. To utilize hard attention in neural networks (Luong et al., 2015) proposed *global attention* and *local attention*. Global attention is similar to soft attention, and it computes attention by extracting the average of the entire input space. In contrast, local attention can be viewed as a blend of hard and soft attention, in which only a subset of the image is considered at a time. However, unlike hard attention, local attention is mostly differentiable.

**Input features** can be split into *item-wise* and *location-wise* categories, depending on whether the input feature is a sequence. The item-wise attention requires that the input is a distinct feature, and location-wise requires a sequence of features.

**Input representation** defines the different types of inputs that can be classified through an attention mechanism. The most common one is the *distinctive attention* which involves a single input with a corresponding label. Multiple labelled inputs are categorized through *co-attention*. Input sequences with different abstraction levels are categorized as *hierarchical attention*. Lastly, *self-attention* extracts inputs from the original input sequence without labels.

**Output representation** defines the types of outputs from attention models. Single-output refers to a single feature representation at each time step. Multiple channels represent the input through multiple channels. Lastly, multi-dimensional output computes a feature-wise score vector for feature maps by replacing the weight scores vector with a matrix. This allows the neural network can calculate multiple attention distributions for the same data.

### Interpretability

In addition to providing performance improvements, attention mechanisms can also be used as a tool to explain the neural network decision process, defined as the interpretability of a neural network. Neural networks are generally considered a “black-box” model where the developer does not know why certain decisions or actions are made (Chaudhari et al., 2021). With growing interest in machine learning and its wide application in decision-making processes that affect our lives, research in the interpretability of a neural network is crucial to creating more transparent systems.  However, there is still some conflict if attention can be a valuable metric to determine the iterability of a neural network. (Serrano and Smith, 2020) argued that attention is not an optimal method of identifying which attended elements are responsible for output, but it may still be interpretable in other ways.

## 1.3.9 Attention in Computer Vision

In computer vision, attention mechanisms can be treated as a selection process that weights feature according to the importance of the input. Each input has its own dimensional space, where each dimension defines a different domain of the input. Because of this, attention applied in computer vision can be applied to different domains of the input space, further splitting attention mechanisms into three distinct categories.

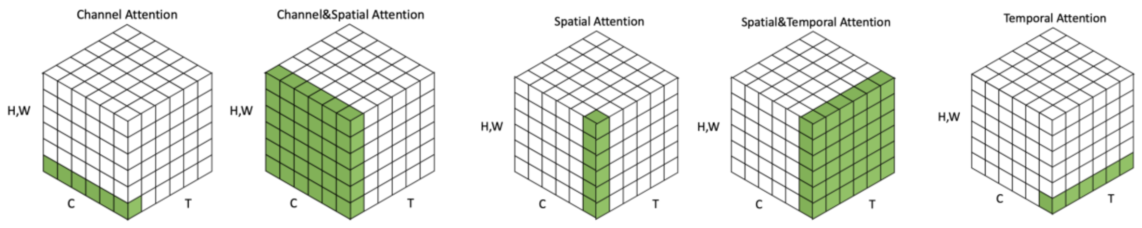


**Figure 1.3: C represents the channel domain, H and W represent spatial domains, and T represents the temporal domain. (Guo et al., 2015)**

The three categories are channel, spatial, and temporal attention. Additionally, two more hybrid categories combine the benefits of the previous attention methods: channel & spatial attention and spatial & temporal attention. Temporal attention will not be discussed as the project focuses on detection from static images and not sequenced data. However, it is important to highlight its existence for future project ideas.

### Channel Attention

During the convolution stage described in section 1.3.4, convolution filters construct feature maps based on the learned weights from the filters. Each feature map learns a different representation of the input, and the number of channels represents the number of convolutional filters. Additionally, we know that feature maps can have varying levels of importance. For example, a feature map that contains where the edge of an image is possibly more important than a feature map that learned to detect a part of the background. Hence, channel attention aims to focus attention on feature maps with a higher level of importance. This is done by adaptively scaling the weight of each channel, thus defining “what to pay attention to”(Guo et al., 2015).

#### Spatial Attention

In computer vision tasks, certain image regions can have more importance than others. For example, if we wanted to detect if a face is present, only the parts of the image that may contain a face are useful. Therefore, applying the face detection algorithm to the global image feature vector may experience poor results due to irrelevant regions. Spatial attention aims to solve this issue by not considering each region in the image to have equal importance and direct attention to the semantic-related regions (Chen et al., 2016). In other words, spatial attention defines “where to pay attention” by learning the positions of the image that should be focused on

### 1.3.10 Emotion Recognition Datasets

It is necessary to have datasets with emotions that are correctly ladled and contain enough data to train the model optimally. Since we aim to classify emotions from static images, the images need to show a variety of expressions and reflect on how people display emotions in real situations. There are two ways to acquire emotion images: from natural sources such as surveillance footage (referred to as “unposed” expressions) or images taken in a lab setting where actors are asked to ‘pose’ expressions in front of a camera. Datasets composed of “unposed” expressions are preferred as they are better representations of real life. On the other hand, “posed” datasets are easier to attain and can be genuine if skilled actors are hired. This project will utilize the three most popular datasets, CK+, JAFFE, and FER2013.

**Extended Cohn-Kanade Dataset** CK+ dataset (Lucey et al., 2010) is an extension of the CK dataset. It contains 593 video sequences and still images of eight facial emotions Neutral, Angry, Contempt, Disgusted, Fearful, Happy, Sad, and Surprised. The data set has 123 subjects, and the facial expressions are posed in a lab. The subjects involved are male and female, with a diversity split of 81% Euro-American, 13% Afro-American, and 6% other.

**JAFFE Dataset** (Lyons et al., 1998) consists of 213 images of different facial expressions from ten different Japanese female subjects. Each subject was asked to pose seven facial expressions (6 basic and neutral). The images were annotated by 60 annotators, where each image has an average semantic rating on each of the facial expressions. The highest average was used as the emotion class for the image.

**FER2013 Dataset** (Carrier et al., 2013) was introduced at the International Conference on Machine Learning (ICML) in 2013 for a Keggle competition. The training set consists of

28,709 examples, and the public test set consists of 3,589 examples. The samples in the dataset differ in age, race, and facial direction, which closely mimics the real world. The human performance on this dataset is estimated to be 65.5 % (Khaireddin and Chen, 2021). Hence, it is widely used as a benchmark for emotion recognition models.

### 1.3.11 Face Detection

In FER, the first pre-requisite step is face detection, which involves detecting the face in the image and removing the insignificant background pixels. Face detection methods aim to find a bounding box where the face is located, and the coordinates of the bounding box can be used to extract the face from the image. Without this step, unwanted features in the image may be extracted and classified along with important information resulting in errors. However, detecting a face is quite complex as the human faces can differ in size, shape, and photos can be taken at different angles.

Initially, facial recognition was done via algorithms, with the most common solution being the “Viola-Jones algorithm” (Viola and Jones, 2004) which is still widely used for simple facial detection tasks. The algorithm has rapid image processing and high accuracy on images where the subject faces the camera however it is less effective on images where the subject is not facing the camera.

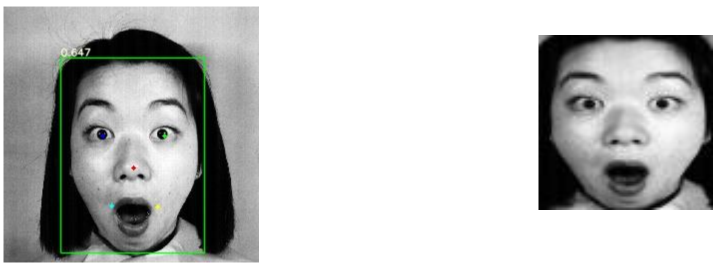
Research from the algorithm approaches showed that facial recognition could be done using standard object recognition approaches via deep CNNs. Object detection approaches are split into two categories: two-stage and one-stage object detector methods. The most popular two-stage methods are the RCNN methods (Girshick et al., 2013) which offer good performance but suffer from long latency and slow computation. Because of this, one-stage object detector architectures were adapted to achieve similar levels of accuracy but with significantly better performance, with the best performing method being the YOLO detector (Bochkovskiy et al., 2020). Because of this, (Qi et al., 2021) adapted the YOLO algorithm for facial recognition achieving state-of-the-art results on the WiderFace dataset (a standard benchmark for facial recognition software) (Yang et al., 2015).

# Chapter 2
# Methods

## 2.1 Pre-Processing the Data

### Face Extraction

The images are loaded into the raw data folder, with each dataset containing a folder for each emotion class. The data is then processed by running facial detection on each image which finds the region where the face is present (the bounding box) and saves the cropped face into the processed data folder alongside the samples class.



*a) Bounding box output* *b)re-sized output used for training*

**Figure 2.1: Example output of the face extraction process**

Facial detection methods were discussed in section 1.3.11. Utilizing the research, the program uses the discussed state-of-the-art facial detector built on top of YOLO5. The paper's authors published their code and made it free to use for other applications (Qi et al., 2021). The repository shares five pre-trained networks that vary in the number of parameters. The higher the number of parameters, the higher accuracy of the face detector, but it comes at a trade of computational speed. After experimenting on a small section of the dataset, “yolov5s” weights were chosen as then the model was always able to detect the face at high speed. Lastly, the original code was modified to add the functionality of cropping the face out of the original image and saving it to a file. The function included a check for the confidante rating, so any region below 60% was not saved, and included a function to reshape the image to specified square dimensions. Since faces bounding boxes can have different proportions, cropped faces must be re-sized so they all have the same size. Images with varying sizes will cause issues when training a deep learning model due to different input tensors hence this is an important step.

### Data Augmentation

Data augmentation was used in the pre-processing stage when the images were loaded into the memory. Possible augmentations considered for the projects were zoom, rotation, width and height shift, shear, and horizontal flip. Colour augments were not considered as the images in the three datasets were all grayscale hence applying colour arguments just added extra noise for the training process rather than resolving the issue of overfitting. Through experimentations, width and height shifts and rotation made the models underfit the data because they have excluded essential parts of the face needed for the classification. On the other hand, horizontal flip and zoom augment improved the accuracy of the models.

## 2.2 Libraries

**TensorFlow** (Zhang et al., 2020) is an end-to-end open-source platform for machine learning. The models within TensorFlow are built using a high-level intuitive library Keras. The main advantage of Keras is that it was designed to make the process of creating machine learning models simple, readable and concise. They achieved this by making a simple API that minimizes the number of actions for common use cases and provides clear error messages. Despite Keras API being designed to be simple, it still offers all the required functionality of more complex machine learning libraries such as PyTorch. In reality, PyTorch and TensorFlow have the same functionality for the scope of this project hence it is up to personal preference which library should be used.

TensorFlow also offers a range of useful libraries. The ones utilized for this project were: tf.models,  tf.math and TensorBoard. Firstly, Tf.models is a library used to compile models, view information about the model such as its structure, train the models and evaluate them.  Secondly, Tf.maths provides several operations that can be used to perform standard math computations on tensor segments. This library created a confusion matrix for the evaluation stage, and its math operations were used to construct attention modules. Lastly, TensorBoard is a tool for storing and viewing information about the model and its training process. In the project, it was used to show loss and accuracy graphs over training and validation data.

**Matplotlib** (Hunter, 2007)is a comprehensive library for creating visualizations in Python. The tool can display figures, tables, and graphs while providing a range of features to style each output. Utilizing this library inside a Jupyter notebook made it easy to display essential information at each stage of the machine learning workflow, from displaying information about the datasets and their contents to displaying evaluation metrics after the training was completed.

**Scikit-learn** (Pedregosaet al., 2011) is a machine learning library designed explicitly for python. In the project, the library displayed evaluation metrics that were missing in TensorFlow. This included the precision, recall, and F1-score values for each emotion class.

## 2.3 CNN Models

The design of the CNN models architecture is a crucial factor in improving the performance of a deep learning network. Innovations in CNNs include a wide range of methods such as modification of processing units, parameter optimization strategies, design patterns, and connectivity of different layers. The first successful CNN used for image classification and detection was AlexNet  (Krizhevsky et al., 2012), which showed incredible performance in ImageNet classification. Since then, critical improvements in CNN performance have occurred mainly due to modifying the model structure and developing novel blocks (Alzubaidi et al., 2021)(Khan et al., 2020).

### 2.3.1 VGGNet

In 2014 (Simonyan and Zisserman, 2014) published Visual Geometry Group (VGGNet), which showed that it is possible to increase the depth of a CNN through small-sized kernels. The paper showed significant improvement to prior research by pushing the depth of the network to  16–19 layers. Additionally, the model introduced simple and sequential architecture upon which most new models are built.

#### Architecture

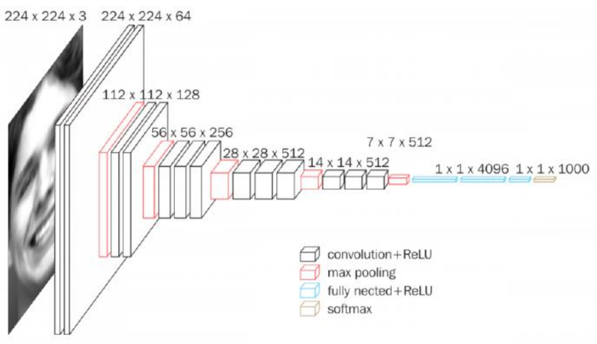


**Figure 2.1 A visualization of the VGG16 architecture**

The image is passed through a stack of convolutional layers with kernel size (3x3). This kernel size is the smallest size, which can capture the idea of left/right, up/down, and centre. The convolution stride is fixed to 1, and padding is set to “same” so that the spatial resolution is preserved after the convolution. Max-pooling is applied with a stride of two after a set number of convolutions to downsample the output of the convolution layer and train the CNN to be spatially invariant. Lastly, a stack of convolutional layers (which have different depths in different architectures) is followed by three fully connected layers: the first two have 4096 channels each, and the third contains seven channels (one for each class). The final layer is the soft-max layer and computes the output emotion class.

| Layer Name | VGG16 | VGG19 |
|---|---|---|
| Conv Block 1 | [3x3,128] x2 | [3x3,128] x2 |
| | MaxPooling | |
| Conv Block 2 | [3x3,128] x2 | [3x3,128] x2 |
| | MaxPooling | |
| Conv Block 3 | [3x3,256] x3 | [3x3,256] x4 |
| | MaxPooling | |
| Conv Block 4 | [3x3,512] x3 | [3x3,512] x4 |
| | MaxPooling | |
| Conv Block 5 | [3x3,512] x3 | [3x3,512] x4 |
| | MaxPooling | |
| FC Layers | FC-4096 | |
| | FC-4096 | |
| Output | FC-7 | |
| | Soft-Max | |

**Table 2.2 CNN Configurations of VGG16 and VGG19 (Simonyan and Zisserman, 2014)**

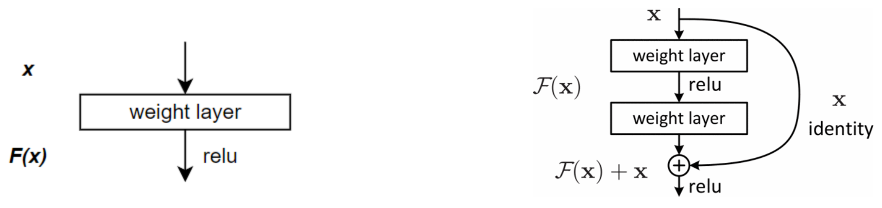
The configuration used for VGG16 and 19 models is described in Table 2.2. The architecture is split into five blocks. The width of the convolution layers (the number of channels) in the first block is 64, and the width increases by a factor of 2 after each max-pooling layer until it reaches 512.

## 2.3.2 ResNet

In 2015 (He et al., 2015) publicized ResNet, which introduced the concept of residual learning and proposed their computational block, the “residual” block. ResNet aimed to solve the issues found in deep CNN architectures: the vanishing gradient and the degradation problem (Glorot and Bengio, 2010). The vanishing gradient problem was already addressed via normalized initialization and batch normalization (Ioffe and Szegedy, 2015) hence ResNet's main aim was to solve the degradation problem. The problem was discovered when deeper networks could converge accuracy at the deeper layers was saturated (the networks learn everything before reaching the final layer) and therefore degraded (He and Sun, 2015). The deeper the architecture, the more saturated the accuracy and higher degradation. The degradation problem was therefore addressed with residual learning.

### Residual Learning

In deeper networks, the deep layers cannot fit a desired underlying mapping required to pass the result to the output. Instead, the layers fit a residual mapping as the network found them easier to optimize.


x
weight layer
F(x)
relu
x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

*a)Desired Mapping* *b)Residual Mapping*

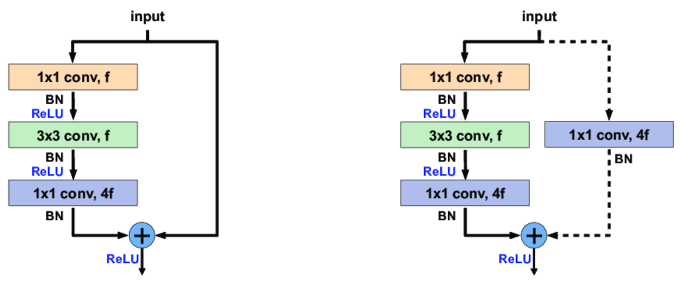
**Figure 2.3: Desired Mapping and Residual Mapping Diagram (He et al., 2015)**

### Shortcut Connection

Residual mapping is formulated with shortcut connections that enable models to skip one or more layers. Shortcut connections enable the model to skip layers that hurt the model's performance, making the model easier to optimize. In ResNet, the shortcut connections perform identity mapping, and their outputs are added to the outputs of the stacked layers. This approach allows easy flow of data between the layers without harming the learning ability of the model.

### Residual Block

In the residual mapping formula $F(x) = F(x) - x$ , $F,$ and $x$ must be equal in dimensions. If that is not the case, we can perform a linear projection in the shortcut connection to match the dimension. Because of this, ResNet uses two blocks in the implementation: the identity block, which performs the standard residual mapping, and the projection block, which includes the linear projection in the shortcut.



*a)Identity block* *b)Projection block*

**Figure 2.4: Residual Blocks for ResNet (Passos et al., 2018)**

Figure 2.4 shows the architecture of the two blocks. $f$ defines the number of filters (times by four on the last convolution), $BN$ is batch normalization, and $conv$ stands for convolution.

### Architecture

The ResNet model is split into five layers inspired mainly by the VGG architecture. After each layer, the feature map size is halved hence the number of filters is doubled to preserve the time complexity per layer. The downsampling of convolution layers is applied directly rather than through a pooling operation by applying a stride of 2 to the initial residual block. Additionally, the first residual block of each layer is the bottleneck block due to the dimension change between the layers. The network ends with a global average pooling layer and a 7-channel fully connected layer (one for each class).

| Layer Name | ResNet 50 | ResNet 101 | ResNet 152 |
|---|---|---|---|
| Conv Block 1 | Conv 7x7, 64, Stride 2<br>3 x 3 Max Pool, Stride 2 | | |
| Conv Block 2 | $\begin{bmatrix}1x1, 64\\3x3, 64\\1x1, 256\end{bmatrix}\times 3$ | $\begin{bmatrix}1x1, 64\\3x3, 64\\1x1, 256\end{bmatrix}\times 3$ | $\begin{bmatrix}1x1, 64\\3x3, 64\\1x1, 256\end{bmatrix}\times 3$ |
| Conv Block 3 | $\begin{bmatrix}1x1, 64\\3x3, 64\\1x1, 256\end{bmatrix}\times 4$ | $\begin{bmatrix}1x1, 64\\3x3, 64\\1x1, 256\end{bmatrix}\times 4$ | $\begin{bmatrix}1x1, 64\\3x3, 64\\1x1, 256\end{bmatrix}\times 8$ |
| Conv Block 4 | $\begin{bmatrix}1x1, 64\\3x3, 64\\1x1, 256\end{bmatrix}\times 6$ | $\begin{bmatrix}1x1, 64\\3x3, 64\\1x1, 256\end{bmatrix}\times 23$ | $\begin{bmatrix}1x1, 64\\3x3, 64\\1x1, 256\end{bmatrix}\times 36$ |
| Conv Block 5 | $\begin{bmatrix}1x1, 64\\3x3, 64\\1x1, 256\end{bmatrix}\times 3$ | $\begin{bmatrix}1x1, 64\\3x3, 64\\1x1, 256\end{bmatrix}\times 3$ | $\begin{bmatrix}1x1, 64\\3x3, 64\\1x1, 256\end{bmatrix}\times 3$ |

**Table 2.5: ResNet 50 , 101 , 152 Architectures (He et al., 2015)**

Table 2.5 shows the difference between ResNet50, 101 and 152. The change between them is the number of residual blocks used at each convolution layer. Because of this, the code implementation for ResNet was split into three parts to avoid repeating the code. Firstly, the residual block function allows switching between identity and projection blocks Appendix 2.2.1. Secondly, A group of residual blocks function calls the residual block function to create a set number of groups of residual blocks at each convolution layer Appendix 2.2.2. Lastly, the main ResNet model function calls the group of residuals function five times, specifying the number of blocks at each layer as described in Table 2.5 (Appendix 2.2.3).

## 2.3.3 ResNet V2

After the release of ResNet, it was discovered that the degradation problem was still present when the depth of the network exceeded 200 layers. The degradation problem inspired the development of ResNet V2 (He et al., 2016), which fully solved the issue of both the vanishing gradient and the degradation problem by implementing pre-activations in the residual blocks. The new version of ResNet increased accuracy for ultra-deep networks exceeding  1001 layers.

### Improved Residual Block

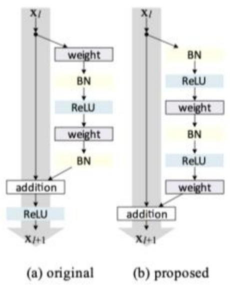


**Figure 2.6: Comparison between the original residual block and the newly proposed one (He et al., 2016)**

Figure 2.6 shows the new proposed residual block and how it differs from the old one. The grey arrows show the easiest path for the information to propagate, which inspired the improvement. In the older implementation of ResNet, ReLU activation is applied after the addition operations, resulting in some loss of information between the layers, which can only be witnessed on ultra-deep networks. Residual blocks in ResNetV2 use pre-activation, which means placing batch normalization and activation before the convolution layer. The result of the addition is then directly passed into the next layer. The input is preserved with no consequence since batch normalization, and RELU always comes before the convolution. The new residual block enables the free flow of information between the layers removing the degradation problem.

### Architecture

ResNetV2 modifies the original residual block but does not change the module's logic and can still be utilized in the same architecture as the original ResNet described in table 2.5. However, the goal of the block was for it to be utilized for ultra-deep networks starting from a depth of 164 layers to 1001. Unfortunately, due to hardware constraints, the project will focus on testing ResNetV2 in the same depth as the original ResNet to see how pre-activation with attention blocks impacts the model's performance.

## 2.3 Attention Modules

Attention modules are designed to be integrated with CNN models to improve them further. Section 1.3.9 discussed how a CNN model could be improved via the channel and spatial attention relating to computer vision tasks. This section expands on the research and will first discuss how attention is implemented in each module, the benefits of each implementation, and possible improvements. Lastly, the section will show how the attention modules integrate within the implemented CNN architectures.

### 2.3.1 SEN-Net (Squeeze and extraction)

The SEN-Net architecture was the first implementation of channel attention in computer vision tasks. The block improved the representational ability of the network by modelling the interdependencies between the channels of the convolutional features. This is done through a feature recalibration operation split into two sequential operations: squeeze and excitation. The first operation collects the feature maps across the spatial dimensions $H \times W$ to produce a channel descriptor.  The second operation captures channel-wise relationships through a self-getting mechanism based on channel dependence and outputs a vector that contains each channel's importance(excitation) level. This vector is then used to reweight the future maps to generate the output of the SE Block, which can be fed into other layers. The diagram of this operation can be seen in Figure 2.7

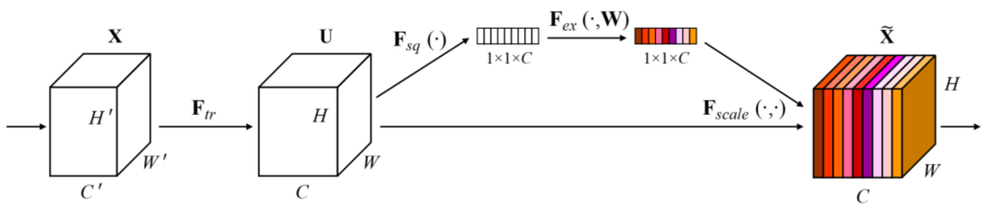


**Figure 2.7: Squeeze and Extraction Attention Block**

The following sections explain the squeeze and extraction operations in more depth and how they are implemented.

## Squeeze

The main purpose of the squeeze operation is to extract global information from each channel in the input filter. Each filter has a set local receptive field, and they cannot use information outside of this region. This is an issue, particularly in the deeper layers where the local receptive field is small. Therefore, global average pooling reduces the $(C \times H \times W)$ filter to $(C \times 1 \times 1)$ to get global spatial information for each channel. The output from the pooling operation is defined as the channel descriptor, and the operation can be formulated as:

$$z_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i,j)$$

The equation describes the process of taking the mean of each channel across the $(H \times W)$ spatial dimensions.

## Excitation

Excitation aims to capture channel-wise dependencies fully via the following gating mechanism:

$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma\big(g(\mathbf{z}, \mathbf{W})\big) = \sigma\big(\mathbf{W}_2 \delta(\mathbf{W}_1 \mathbf{z})\big)$$

The gating mechanism forms a bottleneck via two fully connected layers(FC), $W1$ and $W2$. The input from the squeeze operation is defined as $z$ and is passed onto the first FC layer. The first FC layer is a dimensionality-reduction layer containing a single hyperparameter $r$ which defines the reduction ratio. After the convolution, ReLU($\delta$) activation is used, followed by the second FC layer, which increases the dimension back to the original size. Lastly, A sigmoid($\sigma$) function is used to output the importance(excitation) level for each channel. Sigmoid activation is used instead of SoftMax as it allows multiple channels to have higher importance. For this reason, sigmoid activation is used for multi-label classification.

The output from the excitation operation is then used to rescale the original feature map through a channel-wise multiplication which is possible as s is a scalar.

$$\tilde{\mathbf{x}}_c = \mathbf{F}_{\text{scale}}(\mathbf{u}_c, s_c) = s_c \mathbf{u}_c$$

### Reduction Ratio

The reduction ratio r varies the capacity and computational cost of the squeeze and extraction blocks in the model. A set of experiments was executed using ResNet 50 as the backbone to find the optimal value of $r$. The experiments showed that the optimal value of $r$ was 16, which showed good accuracy without significantly increasing the number of parameters. Accuracy graphs for the experiments are shown in Appendix C.3.6.

| Reduction Ratio | Number of Parameters | Accuracy |
|---|---|---|
| 4 | **33.56M** | 87.2637% |
| 8 | 28.53M | 88.4615% |
| 16 | 26.02M | **89.5604%** |
| 32 | 24.76M | 87.9121% |

**Table 2.8: Effect of the Change of Reduction Ratio on SEN-Net Attention module**

## 2.3.2 ECA-Net

ECA-Net (Wang et al., 2020) was developed to improve channel attention used in SEN-Net. In SEN-net, the excitation module uses dimensionality reduction via two FC layers to extract channel-wise relationships. The channel features are mapped into a low-dimensional space and then mapped back, making the channel connection and its weight indirect. Consequently, this destroys the direct connections between the channel and its weight, reducing the model's performance. Furthermore, empirical studies proved that the operation of dimensional reduction is inefficient and unnecessary for capturing dependencies across all channels(Wang et al., 2020). The ECA-Net goal was to solve the issue of dimensional reduction while improving the efficiency of the excitation operation by introducing an adaptive kernel size within its excitation operation.

### Efficient Channel Attention (ECA) Module

ECA module follows the same architecture as SEN-Net. Firstly, it uses the squeeze operation, which collects the feature maps across the spatial dimensions via the global average pooling operation. Secondly, it executes the excitation operation by performing a 1D convolution of kernel size $k$, where the value of $k$ is adaptively changed based on the number of channels.

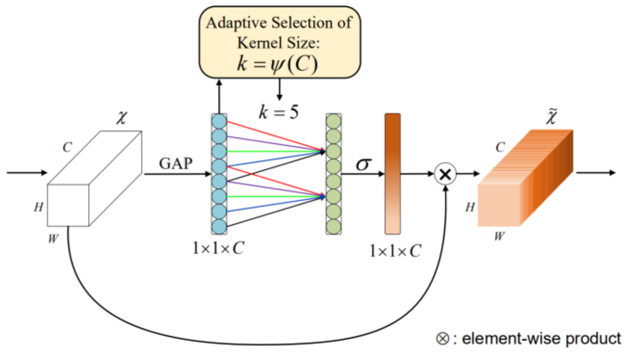


**Figure 2.9: Efficient Channel Attention Module (Wang et al., 2020)**

ECA captures channel-wise relationships by considering every channel and its $k$ neighbours. Therefore, instead of considering all relationships that may be direct or indirect, an ECA block only considers direct interaction between each channel and its k-nearest neighbours to control the model's complexity. Subsequently, ECA-Net improves the performance of the channel attention module by extracting channel relationships via a 1D convolution and not a dimensional reduction operation. It also creates a more efficient module due to the adaptive kernel size being used within the 1D convolution.

### Adaptive Kernel Size

The kernel size $k$ for the 1D convolution kernel is proportional based on the number of channels in the filter map. The authors of the module state that a linear function was too limited hence they proposed a non-linear function based on the idea that the number of channels always relates to a power of 2. As a result, they proposed the following relationship between the kernel size and the number of channels.

$$C = \phi(k) = 2^{(\gamma * k - b)} \tag{1}$$

Deriving formula (1), the value of $k$ can be calculated using the following formula:

$$k = \psi(C) = \left| \frac{log_2(C)}{\gamma} + \frac{b}{\gamma} \right|_{odd}$$

where $|\mathrm{x}|_{odd}$ rounds to the nearest odd number of $x$. $y$ and $b$ are constants and are set to 2 and 1, respectively. Through this equation, filter maps with more channels have a longer range of interactions.

To confirm that the adaptive kernel size was the best option, serval tests were executed to measure the accuracy of the models on different values of $k$. Comparing the results, the adaptive kernel size achieved higher accuracy than the rest, and hence it was used for the rest of the project. Unfortunately, it did so at a higher number of parameters, but the module is still more efficient than SEN-Net while achieving higher accuracies. The output of training and validation accuracy graphs can be found in Appendix C.3.7

| Kernel Size | Number of Parameters | Accuracy |
|---|---|---|
| 1 | 23.50M | 88.46% |
| 3 | 23.53M | 89.65% |
| 5 | 23.56M | 89.11% |
| 7 | 23.59M | 87.36% |
| 9 | 23.62M | 88.56% |
| **Adaptive** | **23.65M** | **90.23%** |

**Figure 2.10: Effect of the Change of Kernel Size ECA Attention module**

## 2.3.3 CBAM

The last attention module implemented in the paper is the CBAM: Convolutional Block Attention Module (Woo et al., 2018). CBAM proposed utilizing both spatial and channel attention to improve the model's performance, unlike the previous attention modules, which only utilised channel attention. The motivation behind the paper stemmed from the fact that convolution operations extract informative features by cross-channel and spatial information together. Therefore, emphasising meaningful features along both dimensions should achieve better results.

### Implementation

CBAM two attention modules are computed sequentially as generating an attention map sequentially outputs a finer attention map than execution in parallel. The channel attention module outputs a channel attention map $M_c$ of dimensions $(C \times 1 \times 1)$ and spatial attention, the module outputs a 2D spatial attention map $M_s$ of dimensions $(1 \times H \times W)$. This is summarised in equation (1):

$$\begin{aligned} \mathbf{F}' &= \mathbf{M_c}(\mathbf{F}) \otimes \mathbf{F} \\ \mathbf{F}'' &= \mathbf{M_s}(\mathbf{F}') \otimes \mathbf{F}' \end{aligned} \tag{1}$$

Chanel and spatial attention maps are applied to the original input space via channel-wise multiplications described by the symbol ⊗. the sequential operation is shown in Figure 2.11.

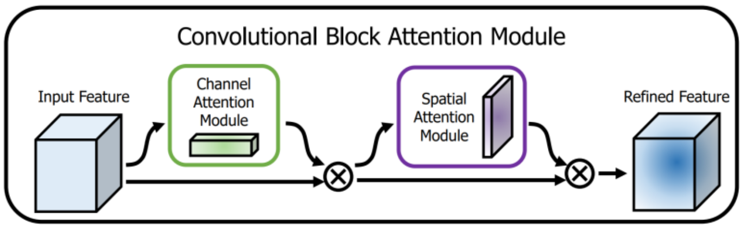


**Figure 2.11: CBAM Attention Block (Woo et al., 2018)**

Additionally, the authors have stated that applying the channel attention module first achieves slightly better performance than the counterpart. However, no matter the order, the model still outperforms channel attention only models showing that utilizing both attentions is crucial while the arrangement strategy can further improve the performance. The following sections explain the implementations of the channel and spatial attention modules.

### Channel Attention Module

Inspired by the implementation of channel attention from SEN-Net(Hu et al., 2018), CBAM channel attention consists of squeeze and excitation operations. However, CBAM modifies the original squeeze operation from SEN-net to include average and max pooling to capture channel-wise dependencies. The idea behind utilizing both steamed from the fact that all spatial regions contribute to the output in average pooling, whereas max-pooling considers the maximum values only. Consequently, combining both improves the representation power of relationships between channels.

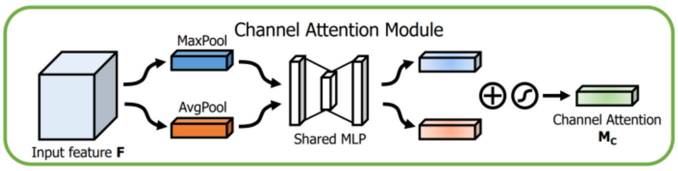


**Figure 2.11: CBAM Chanel Attention Module (Woo et al., 2018)**

As seen in figure 2.11, the two pooling operations are used simultaneously. They are passed to a shared network consisting of two fully connected layers ($W_1$and $W_2$) which perform the excitation operation (following the exact implementation from SEN-Net). After the shared network is applied to the output of each pooling operation, the feature vectors are merged using element-wise summation. The final output is passed through the sigmoid activation layer. In short, the channel attention is computed as:

$$\begin{aligned} \mathrm{M_c(F)} &= \sigma\left(MLP\big(AvgPool(\mathrm{F})\big) + MLP\big(MaxPool(\mathrm{F})\big)\right) \\ &= \sigma\left(\mathrm{W_2}\left(\mathrm{W_1}\left(\mathrm{F^c_{avg}}\right)\right) + \mathrm{W_2}\left(\mathrm{W_1}\left(\mathrm{F^c_{max}}\right)\right)\right) \end{aligned}$$

## Reduction Ratio

Like in SEN-Net, the reduction ratio r allows us to vary the capacity and computational cost of the channel attention block.

| Reduction Ratio | Number of Parameters | Accuracy |
|---|---|---|
| 4 | **33.57M** | 90.46% |
| 8 | 28.54M | 89.01% |
| **16** | 26.02M | **91.21%** |
| 32 | 24.77M | 90.66% |

**Table 2.8: Effect of the Change of Reduction Ratio on SEN-Net Attention module**

The experiments showed that the optimal value of $r$ was 16, the same value used for SEN-net. Traning and validation graphs for the experiments are shown in Appendix C.3.8.

## Spatial Attention Module

The design of the spatial attention module follows the same idea as the channel attention module. To generate a 2D spatial attention map, we compute a 2D spatial descriptor that encodes channel information at each pixel over all spatial locations. This is done via applying average-pooling and max-pooling along the channel axis, after which their outputs are concentrated. The motivation behind this was highlighted in the paper “Paying More Attention to Attention” (Zagoruyko and Komodakis, 2016) which stated that pooling along the channel axis effectively detects informative regions.

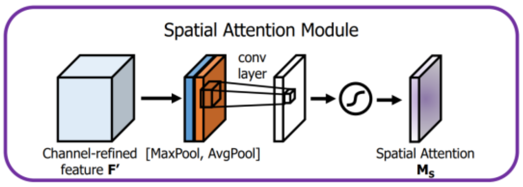


**Figure 2.12: CBAM Spatial Attention Module (Woo et al., 2018)**

The spatial descriptor is then passed to a convolution layer with a kernel size of 7, which outputs the spatial attention map. The choice of the large kernel size was justified by the fact that a large receptive field is needed for deciding spatially important regions. Lastly, the output is passed through a sigmoid function to normalize the output. Therefore, spatial attention is computed as:

$$\mathbf{M_s}(\mathbf{F}) \quad = \sigma\big(f^{7\times 7}([AvgPool(\mathbf{F}); MaxPool(\mathbf{F})])\big)$$

## 2.3.4 Integration with the CNN

### VGGNet

The chosen attention modules are versatile and are designed to be integrated within CNN models. The creators of SEN-Net stated that the SE block could be integrated into standard architectures such as VGGNet by the insertion after the activation layer following each convolution. Through research in the classification of medical images, it was shown that authors had used three different ways to integrate attention in VGGNet: (1) placing attention as described by SEN-Net (Schlemper et al., 2019), (2) placing the attention module before the last fully connected layers (Sitaula and Hossain, 2021) and (3) placing the attention modules at layers 11 and 14 (Wang et al., 2021).

| Method | Number of Parameters | Accuracy |
|---|---|---|
| Method 1 | 39,99M | 89.01% |
| **Method 2** | **39,95M** | **90.11%** |
| Method 3 | 36,81M | 87.91% |

**Table 2.13: Comparing different attention integration methods for VGGNet**

The most optimal configuration was method 2, as shown in table 2.13, and the diagram of the proposed solution is shown in Figure 2.14.

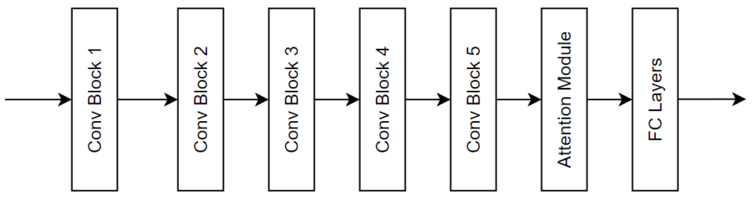


**Figure 2.14: VGGNet with Attention Configuration**

The following configuration had the expected accuracy increase comparable to the one gained from ResNet architectures combined with attention.

### ResNet

Even though ResNet is a more complicated architecture, the creators of SEN-Net provided the most optimal way to integrate their block within the residual block. The integration can be seen in Figure 2.15:

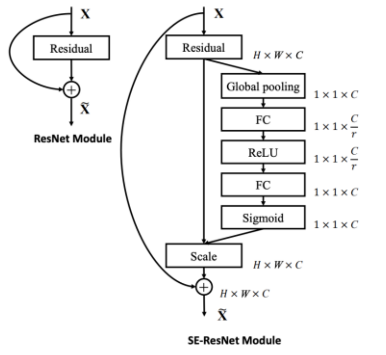


**Figure 2.15: SEN-ResNet Module Diagram (Hu et al., 2018)**

The attention module is added before summation with the identity branch. ECA-Net and CBAM followed the same integration with the residual block.

## 2.4 Experiments

Experiments were executed on the CK+ dataset and the ResNetV1 architecture to execute a range of experiments quickly. Each experiment utilised an early stoppage set at a patience level of 15 epochs and checkpoint to only save the weights at the highest validation accuracy. The train/test split was 80/20%, and a further 30% of the train set was used for the validation set (The motivation behind the split is explained in Appendix C.2.1). Each experiment was executed on a local machine on an Nvidia GTX 1070ti. Furthermore, each experiment was executed serval times to ensure that the results were correct due to the unpredictability of training CNNs. This would usually be accounted for with cross-validation. However, this was not possible due to computational handicap. Furthermore, to confirm the results were correct, the graphs extracted from TenserBoard were inspected, and outlier results were not considered. Accuracy graphs and performance analysis can be found in Appendix C.3.

| **Calibration Parameters** | **Search Space** | **Final Settings** |
|---|---|---|
| Learning Rate | {0.1, 0.001, 0.0001, 0.00001} | 0.0001 |
| Batch Size CK+ | {8, 16, 32, 64} | 16 |
| Batch Size JAFFE | {2, 4, 8, 16} | 4 |
| Batch Size FER2013 | {32, 64, 128, 256} | 64 |
| Activation Functions | {ReLU, ELU, SELU} | ELU |
| Optimizer | {Adam, RSMProp, ADAGrad, SDG} | Adam |
| Pooling | {Max Pooling, Average Pooling} | Average Pooling |
| SEN-Net Reduction Ratio | {4, 8, 16, 32} | 16 |
| ECA-Net Kernel Size | {1, 3, 5, 7, 9, Adaptive} | Adaptive |
| CBAM Reduction Ratio | {4, 8, 16, 32} | 16 |

**Table 2.16: Hyperparameters used for the final evaluation and the corresponding search space**

# Chapter 3
# Results

## 3.1 Evaluation Metrics

The final models will be evaluated by executing the trained models on a test set and recording the model's accuracy on unseen data. The equation for accuracy is specified in equation (1), where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \qquad (1)$$

Each training process is saved to TenserBoard, which records the training and validation sets accuracy and loss throughout the training. TensorBoard can display graphs with the log files and provide an interface that compares training results from different executions. Additionally, a confusion matrix will be used to show how well the model classifies each emotion category and highlight which data samples are misclassified.

## 3.2 Base CNNs

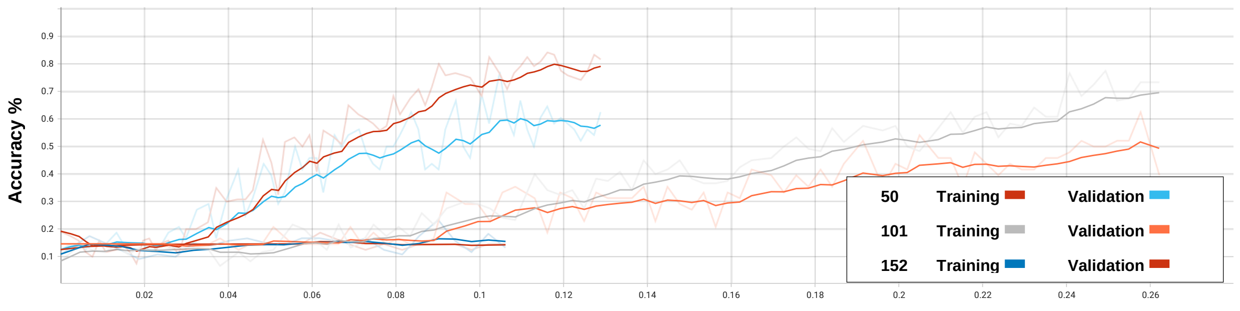
Table 3.1 displays the final evaluation accuracies of the CNN models on the three datasets. The evaluations for CK+ and JAFFE were executed three times to ensure the correctness of the results, as with smaller datasets, there is a fluctuation of evaluation accuracies between the runs. Out of the three executions, the highest value was chosen.

| Architecture | Param | CK+ Accuracy | JAFFE Accuracy | FER2013 Accuracy |
|---|---|---|---|---|
| VGG-16 | 39.92 M | 87.91% | 64.44% | 60.66% |
| VGG-19 | **42.87M** | **90.66%** | **68.89%** | **60.92%** |
| ResNet-50 | 23.49M | 87.91% | **73.33%** | 58.61% |
| ResNet-101 | 42.46M | **88.46%** | 60.00% | 58.67% |
| ResNet-152 | **58.08M** | 85.71% | 15.66% | **59.36%** |
| ResNetV2-50 | 23.48M | 88.46% | **77.78%** | 58.72% |
| ResNetV2-101 | 42.44M | 88.62% | 62.22% | 59.07% |
| ResNetV2-152 | **58.05M** | **89.01%** | 66.67% | **59.40%** |

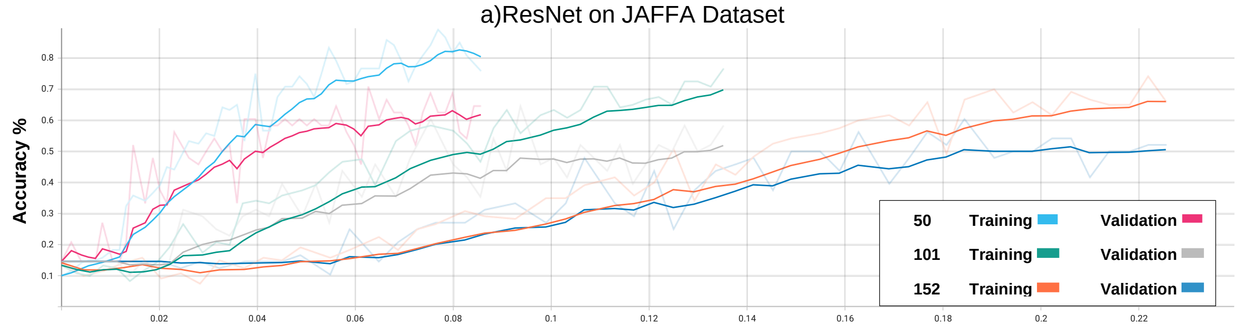
**Table 3.1: Evaluation of CNN architectures on CK+, JAFFE and FER2013**

Analysing the results, we see that VGG-19 achieved the best accuracy on CK+ and FER2013, while ResNetV2-50 achieved the best accuracy on the JAFFE dataset. This was an unexpected result as the initial assumption was that the deeper ResNet models should outperform VGGNet, which was not the case. This may be due to the modification of the activation function from ReLU to ELU in the VGGNet models. This change improved the VGG-19 accuracy from 87.91% to 90.66% on the CK+ dataset, a significantly better improvement than the deeper ResNet models for the same modification. Furthermore, the deeper ResNet models consist of more parameters than VGG-19. Because deep CNNs are designed to be trained on large amounts of data, the layers at the deeper stages cannot learn informative features. Consequently, overfitting occurs.

Another interesting finding was that ResNetV2 performed significantly better than ResNet on the smallest JAFFA dataset. Original ResNet showed degradation in accuracy past depth 101 and could not increase training accuracy past depth 152 on the JAFFE dataset. On the other hand, ResNetV2 was still able to train on the deeper models, and the model of 50 layers performed better than the original ResNet. From these results, there might be a discovery that the new improved residual blocks perform better than the original when less data is present due to the easier flow of information thanks to the pre-activation functions. The new improved residual blocks also perform better than the original on CK+ and FER2013, confirming the initial assumptions. A comparison between relative training graphs of ResNet and ResNetV2 on the JAFFA dataset can be seen in Figure 3.2.



a)ResNet on JAFFA Dataset



b)ResNetV2 on JAFFA Dataset

**Figure 3.2: training and validation accuracy graphs of ResNet and ResNetV2 on the JAFFA dataset**

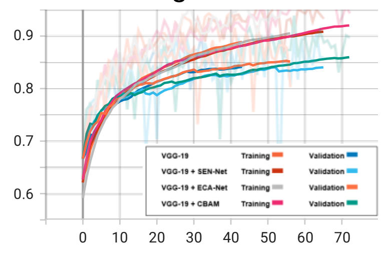
Relative graphs were chosen to separate the ResNet models as the larger models will have a longer computational time. Figure 3.2 shows that ResNetV2 converges to optimal values faster, and the performance degradation in the deeper layers is not as sudden as the original ResNet.

## 3.3 CNN with Attention Accuracy Results

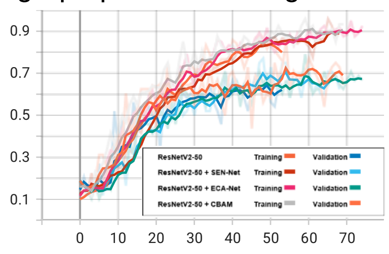
| Model | Param | CK + Accuracy | JAFFE Accuracy | FER2013 Accuracy |
|---|---|---|---|---|
| VGG-16 | 39.92 M | 87.91% | 64.44% | 60.66% |
| VGG-16 + SEN-Net | 39.95M | 88.46% | 68.89% | 63.05% |
| VGG-16 + ECA-Net | 39.92M | 89.01% | 73.33% | 62.72% |
| VGG-16 + CBAM | **39.95M** | **89.56%** | **75.56%** | **63.46%** |
| VGG-19 | 42.87M | 90.66% | 68.89% | 60.92% |
| VGG-19 + SEN-Net | 45.26M | 91.21% | 73.33% | 63.23% |
| VGG-19 + ECA-Net | 45.23M | 91.76% | 75.56% | 63.49% |
| VGG-19 + CBAM | **45.26M** | **92.31% (↑ 1.65%)** | **77.78%** | **64.07% (↑ 3.15%)** |
| ResNet-50 | 23.49M | 87.91% | 73.33% | 58.61% |
| ResNet-50 + SEN-Net | 26.02M | 89.01% | 75.56% | 58.84% |
| ResNet-50 + ECA-Net | 23.65M | 90.11% | 77.78% | 59.73% |
| ResNet-50 + CBAM | **26.02M** | **91.21%** | **82.22%** | **59.90%** |
| ResNet-101 | 42.46M | 88.46% | 60.00% | 58.67% |
| ResNet-101 + SEN-Net | 47.24M | 89.01% | 68.89% | 58.92% |
| ResNet-101 + ECA-Net | 42.81M | 89.56% | 73.33% | 60.15% |
| ResNet-101 + CBAM | **47.24M** | **90.11%** | **75.56%** | **60.92%** |
| ResNet-152 | 58.08M | 85.71% | 15.66% | 59.36% |
| ResNet-152 + SEN-Net | 64,71M | 88.46% | 15.66% | 59.73% |
| ResNet-152 + ECA-Net | 58.60M | 89.56% | 15.66% | 60.92% |
| ResNet-152 + CBAM | **64.71M** | **90.11%** | 15.66% | **61.54%** |
| ResNetV2-50 | 23.48M | 88.46% | 77.78% | 58.72% |
| ResNetV2-50 + SEN-Net | 26.01M | 88.66% | 82.22% | 59.36% |
| ResNetV2-50 + ECA-Net | 23.64M | 88.91% | 82.22% | 59.73% |
| ResNetV2-50 + CBAM | 26.01M | **89.01%** | **84.44%(↑ 6.55%)** | **60.15%** |
| ResNetV2-101 | 42.44M | 88.62% | 62.22% | 59.07% |
| ResNetV2-101 + SEN-Net | 47,22M | 89.01% | 68.89% | 59.73% |
| ResNetV2-101 + ECA-Net | 42.79M | 89.56% | 70.83% | 60.15% |
| ResNetV2-101 + CBAM | 47.22M | **90.66%** | **73.33%** | **60.92%** |
| ResNetV2-152 | 58.05M | 89.01% | 66.67% | 59.40% |
| ResNetV2-152 + SEN-Net | 64.68M | 89.56% | 68.89% | 60.72% |
| ResNetV2-152 + ECA-Net | 58.57M | 89.82% | 73.33% | 61.54% |
| ResNetV2-152 + CBAM | **64.69M** | **90.11%** | **77.78%** | **62.05%** |

**Table 3.3: Evaluation of attention modules on CNN architectures on CK+, JAFFE, and FER2013**

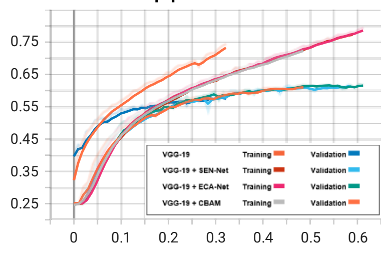
Table 3.3 summarizes the experimental results. The networks with attention outperformed all the baselines significantly, demonstrating that attention can generalise well on various models. Moreover, the addition of attention showed performance improvement across all the datasets displaying that attention could be applied to any problem size. Figure 3.4 shows the accuracy curves of the best performing networks. In each case, attention achieves higher accuracies and shows a smaller gap between training and validation curves than baseline networks. Larger versions of the graph presented in figure 3.4 are shown in appendix C4.1.



a)VGG19 on CK+



b)ResNetV2-50 on JAFFE



c)VGG19 on FER2013

**Figure 3.4: Accuracy curves for the best performing models**

As expected, CBAM had the best improvement in accuracy over the other attention modules due to the application of spatial attention. However, that comes at the cost of a significant overhead in parameters. On the other hand, ECA-Net achieved similar levels of performance increase compared to CBAM while not significantly impacting the memory requirement of each network. However, each attention module caused a significant increase in training time. Figure 3.5 shows the impact the execution time attention modules have on the baseline networks.

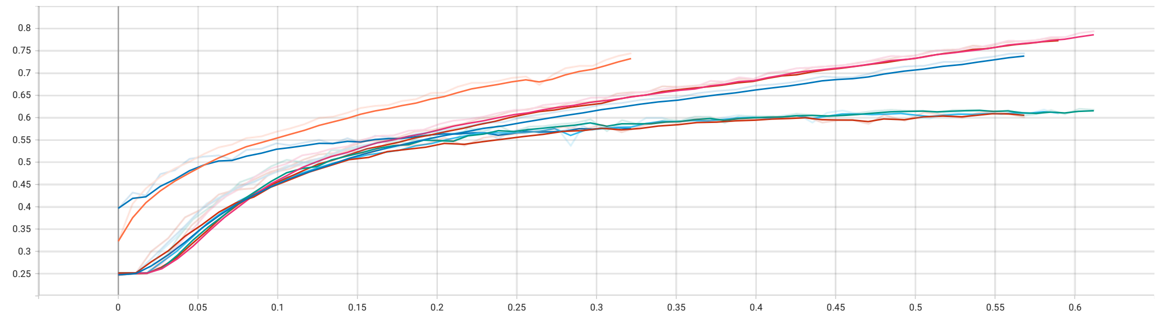


**Figure 3.5: Relative execution time graph of VGG-19 on the FER2013 dataset**

The addition of attention did not change the best performing networks from the baseline CNN comparisons. VGG19 still achieved the best performance on the CK+ and FER2013 datasets, while ResNetV2-50 achieved the best performance on the JAFFA dataset. However, the increase in performance was significantly higher than expected in the FER2013 dataset. Due to the size of the dataset, the expected improvement should have been 1-2% which is the improvement authors of CBAM received on the ImageNet dataset. However, CBAM achieved a performance increase of 3.15% on FER2013, displaying that attention modules can significantly impact the network's performance. Furthermore, the addition of CBAM enabled an increase of 6.55% on the JAFFA dataset, demonstrating the ability of attention modules to improve the network's generalisation ability.

The significant improvements witnessed on the JAFFA dataset can also imply that attention modules perform best on balanced datasets. This is supported by the confusion matrices presented in Appendix C4.1. The JAFFA test set has misclassified data to only one other emotion category. Comparing this to CK+ and FER2013, which both have the issue of data imbalance, this is not the case. The errors in classification are split across a range of emotions which is not ideal. On the other hand, JAFFA is a small dataset, and its test set only consisted of 45 images, so it is hard to compare its result to the other datasets.

# Chapter 4
# Discussion

## 4.1 Conclusions

The project has successfully displayed deep learning models to classify the seven basic human emotions. Firstly, the project discussed the implementation of the chosen CNN models VGGNet, ResNet, and ResNetV2. The proposed CNNs were modified through experimentation by replacing their internal activation function from ReLU to ELU, which significantly improved their performance by solving the bias-shift problem. Additionally, The paper has discovered that new residual blocks presented in ResNetV2 perform significantly better than the original on smaller datasets and show a slight improvement on mid-sized and larger-sized datasets. This may be because of the uninterrupted flow of information between the layers, thanks to the pre-activation functionality.

Secondly, the concept of attention was introduced, which aimed to refine the extracted features and improve the generalisation of the models. The paper showed channel and spatial attention implementation for the facial emotion recognition task. The presented implementations of attention modules were SEN-Net, ECA-Net, and CBAM. SEN-Net and ECA-Net implemented channel attention only, while CBAM utilised both channel and spatial attention to achieve a considerable performance improvement. The proposed attention modules were designed to improve CNNs for the ImageNet classification task. Therefore, the attention module hyperparameters were modified through experimentation to maximize the performance of the models on the tasks of emotion recognition.

The project's main goal was to verify the effect attention has and how it improves the performance of CNNs. Each attention module was combined with the implemented CNNs, and extensive experiments were conducted on three distinct datasets of different sizes (CK+, JAFFE, FER2013). As shown in section 3.3, each attention module outperformed the baseline models on each dataset. Consequently, this showed that attention modules could successfully improve the generalisation ability no matter the problem size. Furthermore, the results proved that the application of both channel and spatial attention achieves the best results as CBAM had the best performance in all the results. Overall attention improved the CK+ baseline model by 1.65%, JAFFE by 6.55%, and FER2013 by 3.15%, overall achieving accuracies of 92.31%, 84.44%, and 64.07%, respectively. The accuracy achieved on the FER2013 dataset is comparable to the human performance of 65.5%, displaying that the project successfully achieved its goals.

## 4.2 Ideas for future work

There is a vast range of opportunities for future work that can be considered for this project. Future projects can address the data imbalance issue presented in the CK+ and FER2013 datasets. The confusion matrices for these datasets have shown insufficient data for the disgust class in the FER2013 dataset, and CK+ has significantly more samples in the natural class. For these reasons, CK+ models tend to classify most images as natural, and in the FER2013 disgust class has the lowest accuracy. Solving this issue may significantly improve the performance of the models. Furthermore, both CK+ and JAFFE datasets do not contain enough samples to utilise hierarchical learning within deep CNN models. Future works can aim to address this problem in two ways: artificially increasing the dataset sizes through data augmentation or transfer learning.

Future works may also focus on the different CNN architectures not considered for this project, such as ResNetX or Inception. It would also be worth looking into improving these architectures for emotion recognition via new activation functions and optimizers. This project displayed a significant performance increase that can be achieved by utilizing ELU over ReLU, and it is worth exploring these options in future projects. Additionally, it would be worth exploring the performance change of emotion recognition in ultra-deep networks (200+ layered networks), which was impossible for this project due to computation constraints.

Subsequently, the current implementation of the CBAM utilises the dimensionality reduction operation to extract the channel-wise relationships. It is worth exploring improvements to the CBAM module by utilising the improved squeeze operation presented by ECA-Net. Implementing an adaptive kernel size may significantly reduce the overhead of the CBAM module and show improvements in the overall performance.

Thirdly, temporal attention could be utilized to expand the project to real-time emotion recognition. Classification of emotion from a sequence of images may show performance improvement as the model can capture more information from the images. The models will be able to learn the transitions from one emotion to another, further distinguishing the salient regions in the samples.

Lastly, future projects may explore the problem of interpretability within CNN's through utilising attention modules. There is currently not enough research in this field, and there are conflicting opinions on whether attention is valuable to determine the interpretability of the network.

## List of References

Albawi, S., Mohammed, T.A. and Al-Zawi, S. 2018. Understanding of a convolutional neural network. *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*. **2018-Janua**, pp.1–6.

Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M. and Farhan, L. 2021. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data 2021 8:1*. **8**(1), pp.1–74.

Aneja, D., Colburn, A., Faigin, G., Shapiro, L. and Mones, B. 2017. Modeling stylized character expressions via deep learning *In*: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Verlag, pp.136–153.

Aneja, D., Colburn, A., Faigin, G., Shapiro, L. and Mones, B. 2016. Modeling Stylized Character Expressions via Deep Learning. *undefined*. **10112 LNCS**, pp.136–153.

Bahdanau, D., Cho, K.H. and Bengio, Y. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Bakker, I., van der Voordt, T., Vink, P. and de Boon, J. 2014. Pleasure, Arousal, Dominance: Mehrabian and Russell revisited. *Current Psychology*. **33**(3), pp.405–421.

Barry-Straume, J., Tschannen, A., Engels, D.W. and Fine, E. 2018. *An Evaluation of Training Size Impact on Validation Accuracy for Optimized Convolutional Neural Networks* [Online]. [Accessed 8 April 2022]. Available from: https://scholar.smu.edu/datasciencereview/vol1/iss4/12.

Basha, S.H.S., Dubey, S.R., Pulabaigari, V. and Mukherjee, S. 2020. Impact of fully connected layers on performance of convolutional neural networks for image classification. *Neurocomputing*. **378**, pp.112–119.

BCS 2020. BCS, The Chartered Intstitute for IT Code of Conduct for BCS Members. . (June), pp.1–6.

Bengio, Y., Courville, A. and Vincent, P. 2012. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **35**(8), pp.1798–1828.

Bera, S. and Shrivastava, V.K. 2020. Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification. *International Journal of Remote Sensing*. **41**(7), pp.2664–2683.

Bjorck, J., Gomes, C., Selman, B. and Weinberger, K.Q. 2018. Understanding batch normalization *In*: *Advances in Neural Information Processing Systems*., pp.7694–7705.

Bochkovskiy, A., Wang, C.-Y. and Liao, H.-Y.M. 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection.

Britz, D., Goldie, A., Luong, M.T. and Le, Q. V. 2017. Massive Exploration of Neural

Machine Translation Architectures. *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*., pp.1442–1451.

Carrier, P.-L., Courville, A., Goodfellow, I.J., Mirza, M. and Bengio, Y. 2013. FER-2013 face database. *Universit de Montral*.

Cerf, M., Frady, E.P. and Koch, C. 2009. Faces and text attract gaze independent of the task: Experimental data and computer model. *Journal of Vision*. **9**(12), pp.10–10.

Chaudhari, S., Mithal, V., Polatkan, G. and Ramanath, R. 2021. An Attentive Survey of Attention Models. *ACM Transactions on Intelligent Systems and Technology*. **12**(5), p.33.

Chen, L., Zhang, H., Xiao, J., Nie, L., Shao, J., Liu, W. and Chua, T.S. 2016. SCA-CNN: Spatial and Channel-wise Attention in Convolutional Networks for Image Captioning. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. **2017-January**, pp.6298–6306.

Chollet, Francois and and Others 2015. Keras.

Chu, H.C., Tsai, W.W.J., Liao, M.J. and Chen, Y.M. 2018. Facial emotion recognition with transition detection for students with high-functioning autism in adaptive e-learning. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*. **22**(9), pp.2973–2999.

Clevert, D.A., Unterthiner, T. and Hochreiter, S. 2016. Fast and accurate deep network learning by exponential linear units (ELUs) *In*: *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*.

Cohn, J.F. and Zlochower, A. 1995. A computerized analysis of facial expression: Feasibility of automated discrimination. *American Psychological Society*. **2**(6).

Connor, C.E., Egeth, H.E. and Yantis, S. 2004. Visual Attention: Bottom-Up Versus Top-Down. *Current Biology*. **14**(19), pp.R850–R852.

Corive, R., Douglas-Cowie, E., Tsapatsoulis, N., Votsis, G., Kollias, S., Fellenz, W. and Taylor, J.G. 2001. Emotion recognition in human-computer interaction. *IEEE Signal Processing Magazine*. **18**(1), pp.32–80.

Delong Qi and Weijun Tan, Qi Yao and Jingfeng Liu 2021. YOLO5Face: Why Reinventing a Face Detector.

Desimone, R. and Duncan, J. 1995. Neural mechanisms of selective visual attention. *Annual review of neuroscience*. **18**, pp.193–222.

Domingos, P. 2000. A unified bias-variance decomposition *In*: *Proceedings of 17th international conference on machine learning*. Morgan Kaufmann Stanford, pp.231–238.

Duchi JDUCHI, J. and Singer, Y. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization * Elad Hazan. *Journal of Machine Learning Research*. **12**, pp.2121–2159.

Edwards, J., Jackson, H.J. and Pattison, P.E. 2002. Emotion recognition via facial expression and affective prosody in schizophrenia: a methodological review. *Clinical psychology review*. **22**(6), pp.789–832.

Ekman, P. 1992. An Argument for Basic Emotions. *Cognition and Emotion*. **6**(3–4), pp.169–200.

Galdran, A., Alvarez-Gila, A., Meyer, M.I., Saratxaga, C.L., Araújo, T., Garrote, E., Aresta, G., Costa, P., Mendonçmendonç¸a, A.M. and Campilho, A. 2017. Data-Driven Color Augmentation Techniques for Deep Skin Image Analysis.

Giannopoulos, P., Perikos, I. and Hatzilygeroudis, I. 2018. Deep learning approaches for facial emotion recognition: A case study on FER-2013. *Smart Innovation, Systems and Technologies*. **85**, pp.1–16.

Gifford, C. 2020. The problem with emotion-detection technology – The New Economy. *New Economy*. [Online]. [Accessed 16 January 2022]. Available from: https://www.theneweconomy.com/technology/the-problem-with-emotion-detection-technology.

Girshick, R., Donahue, J., Darrell, T. and Malik, J. 2013. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*., pp.580–587.

Glorot, X. and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks *In*: *Journal of Machine Learning Research* [Online]., pp.249–256. [Accessed 11 April 2022]. Available from: http://www.iro.umontreal.

Glorot, X., Bordes, A. and Bengio, Y. 2011. *Deep Sparse Rectifier Neural Networks* [Online]. JMLR Workshop and Conference Proceedings. [Accessed 27 January 2022]. Available from: https://proceedings.mlr.press/v15/glorot11a.html.

Goodfellow, I.J., Erhan, D., Luc Carrier, P., Courville, A., Mirza, M., Hamner, B., Cukierski, W., Tang, Y., Thaler, D., Lee, D.H., Zhou, Y., Ramaiah, C., Feng, F., Li, R., Wang, X., Athanasakis, D., Shawe-Taylor, J., Milakov, M., Park, J., Ionescu, R., Popescu, M., Grozea, C., Bergstra, J., Xie, J., Romaszko, L., Xu, B., Chuang, Z. and Bengio, Y. 2015. Challenges in representation learning: A report on three machine learning contests. *Neural Networks*. **64**, pp.59–63.

Guo, M.-H., Xu, T.-X., Liu, J.-J., Liu, Z.-N., Jiang, P.-T., Mu, T.-J., Zhang, S.-H., Martin, R.R., Cheng, M.-M., Member, S. and Hu, S.-M. 2015. Attention Mechanisms in Computer Vision: A Survey. . **14**(8).

Han, K., Yu, D. and Tashev, I. 2014. Speech emotion recognition using deep neural network and extreme learning machine *In*: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*., pp.223–227.

Hastie, T., Tibshirani, R. and Friedman, J. 2009. Model Assessment and Selection *In*: Springer, New York, NY, pp.219–259. [Accessed 7 February 2022]. Available from: https://link.springer.com/chapter/10.1007/978-0-387-84858-7_7.

He, K. and Sun, J. 2015. Convolutional neural networks at constrained time cost. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. **07-12-June-2015**, pp.5353–5360.

He, K., Zhang, X., Ren, S. and Sun, J. 2015. Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. **2016-December**, pp.770–778.

He, K., Zhang, X., Ren, S. and Sun, J. 2016. Identity Mappings in Deep Residual Networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. **9908 LNCS**, pp.630–645.

Hinton, G., Srivastava, N. and Swersky, K. 2014. Neural Networks for Machine Learning, Lecture 6.

Hopfield, J.J. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*. **79**(8), p.2554.

Hu, J., Shen, L. and Sun, G. 2018. *Squeeze-and-Excitation Networks*. IEEE Computer Society.

Huang, Y., Yang, J., Liao, P. and Pan, J. 2017. Fusion of Facial Expressions and EEG for Multimodal Emotion Recognition. *Computational Intelligence and Neuroscience*. **2017**.

Hudlicka, E. 2011. Guidelines for Designing Computational Models of Emotions. *International Journal of Synthetic Emotions*. **2**(1), pp.26–79.

Hudlicka, E. 2008. What are we modeling when we model emotion? *In*: *AAAI Spring Symposium - Technical Report* [Online]., pp.52–59. [Accessed 21 January 2022]. Available from: https://www.researchgate.net/publication/221251193_What_Are_We_Modeling_When_We_Model_Emotion/citation/download.

Hunter, J.D. 2007. Matplotlib: A 2D graphics environment. *Computing in Science and Engineering*. **9**(3), pp.90–95.

Ian Goodfellow, Yoshua Bengio and Aaron Courville 2016. *Deep Learning* [Online]. MIT Press. [Accessed 25 January 2022]. Available from: http://www.deeplearningbook.org.

Ioffe, S. and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *32nd International Conference on Machine Learning, ICML 2015*. **1**, pp.448–456.

Jost, T., Ouerhani, N., Wartburg, R. Von, Müri, R. and Hügli, H. 2005. Assessing the contribution of color in visual attention. *Computer Vision and Image Understanding*. **100**(1-2 SPEC. ISS.), pp.107–123.

Kandel, I., Castelli, M. and Popovič, A. 2020. Comparative Study of First Order Optimizers for Image Classification Using Convolutional Neural Networks on Histopathology Images. *Journal of Imaging 2020, Vol. 6, Page 92*. **6**(9), p.92.

Khaireddin, Y. and Chen, Z. 2021. Facial Emotion Recognition: State of the Art Performance on FER2013.

Khan, A., Sohail, A., Zahoora, U. and Qureshi, A.S. 2020. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*. **53**(8), pp.5455–5516.

Kim, H.R., Kim, Y.S., Kim, S.J. and Lee, I.K. 2018. Building emotional machines: Recognizing image emotions through deep neural networks. *IEEE Transactions on Multimedia*. **20**(11), pp.2980–2992.

Kingma, D.P. and Ba, J.L. 2014. Adam: A Method for Stochastic Optimization. *3rd*

*International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Klambauer, G., Unterthiner, T., Mayr, A. and Hochreiter, S. 2017. Self-normalizing neural networks *In*: *Advances in Neural Information Processing Systems*., pp.972–981.

Krieger, G., Rentschler, I., Hauske, G., Schill, K. and Zetzsche, C. 2000. Object and scene analysis by saccadic eye-movements: An investigation with higher-order statistics. *Spatial Vision*. **13**(2–3), pp.201–214.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*. **25**.

Lecun, Y., Bengio, Y. and Hinton, G. 2015. Deep learning. *Nature*. **521**(7553), pp.436–444.

Lee, J., Kim, Sunok, Kim, Seungryong and Sohn, K. 2020. Multi-Modal Recurrent Attention Networks for Facial Expression Recognition. *IEEE Transactions on Image Processing*. **29**, pp.6977–6991.

Li, B., Ren, H., Jiang, X., Miao, F., Feng, F. and Jin, L. 2021. SCEP - A New Image Dimensional Emotion Recognition Model Based on Spatial and Channel-Wise Attention Mechanisms. *IEEE Access*. **9**, pp.25278–25290.

Li, S. and Deng, W. 2020. Deep Facial Expression Recognition: A Survey. *IEEE Transactions on Affective Computing*.

Li, W., Liu, K., Zhang, L. and Cheng, F. 2020. Object detection based on an adaptive attention mechanism. *Scientific Reports 2020 10:1*. **10**(1), pp.1–13.

Lu, J., Xiong, C., Parikh, D. and Socher, R. 2016. Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. **2017-January**, pp.3242–3250.

Lucey, P., Cohn, J.F., Kanade, T., Saragih, J., Ambadar, Z. and Matthews, I. 2010. The extended Cohn-Kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, CVPRW 2010*., pp.94–101.

Luo, Y., Zhu, L.Z. and Lu, B.L. 2019. A GAN-Based Data Augmentation Method for Multimodal Emotion Recognition *In*: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* [Online]. Springer Verlag, pp.141–150. [Accessed 19 April 2022]. Available from: https://link.springer.com/chapter/10.1007/978-3-030-22796-8_16.

Luong, M.T., Pham, H. and Manning, C.D. 2015. Effective approaches to attention-based neural machine translation *In*: *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*., pp.1412–1421.

Lyons, M., Kamachi, M. and Gyoba, J. 1998. The Japanese Female Facial Expression (JAFFE) Dataset.

Majumder, N., Poria, S., Hazarika, D., Mihalcea, R., Gelbukh, A. and Cambria, E. 2019. DialogueRNN: An attentive RNN for emotion detection in conversations *In*: *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019* [Online]., pp.6818–6825. [Accessed 19 April 2022]. Available from: www.aaai.org.

Manas Sambare 2013. FER-2013 | Kaggle. [Accessed 27 April 2022]. Available from: https://www.kaggle.com/datasets/msambare/fer2013.

Mao, Q., Zhu, Q., Rao, Q., Jia, H. and Luo, S. 2019. Learning Hierarchical Emotion Context for Continuous Dimensional Emotion Recognition from Video Sequences. *IEEE Access*. **7**, pp.62894–62903.

Mehrabian, A. 1968. Communication without words. *Psychol. Today 2(4)*., pp.53–56.

Mikołajczyk, A. and Grochowski, M. 2018. Data augmentation for improving deep learning in image classification problem. *2018 International Interdisciplinary PhD Workshop, IIPhDW 2018*., pp.117–122.

Minaee, S., Minaei, M. and Abdolrashidi, A. 2019. Deep-Emotion: Facial Expression Recognition Using Attentional Convolutional Network. *Sensors*. **21**(9).

Mohammad, S.M. 2021. Ethics Sheet for Automatic Emotion Recognition and Sentiment Analysis.

Nair, V. and Hinton, G.E. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. *undefined*.

Niu, Z., Zhong, G. and Yu, H. 2021. A review on the attention mechanism of deep learning. *Neurocomputing*. **452**, pp.48–62.

Osuna, E., Rodríguez, L.F., Gutierrez-Garcia, J.O. and Castro, L.A. 2020. Development of computational models of emotions: A software engineering perspective. *Cognitive Systems Research*. **60**, pp.1–19.

Passos, L.A., Pereira, C.R., Rezende, E.R.S., Carvalho, T.J., Weber, S.A.T., Hook, C. and Papa, J.P. 2018. *Parkinson disease identification using residual networks and optimum-path forest*.

Pedamonti, D. 2018. Comparison of non-linear activation functions for deep neural networks on MNIST classification task.

Pedregosa, F; Varoquaux, G; Gramfort, A; Michel, V; Thirion, B; and Grisel, O. and Blondel, . and Prettenhofer, P., and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and, Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E., Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, É. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. **12**(85), pp.2825–2830.

Perez, L. and Wang, J. 2017. The Effectiveness of Data Augmentation in Image Classification using Deep Learning.

Petrantonakis, P.C. and Hadjileontiadis, L.J. 2010. Emotion recognition from EEG using higher order crossings. *IEEE Transactions on Information Technology in*

*Biomedicine*. **14**(2), pp.186–197.

Pinto, Y., van der Leij, A.R., Sligte, I.G., Lamme, V.A.F. and Scholte, H.S. 2013. Bottom-up and top-down attention are independent. *Journal of Vision*. **13**(3), pp.16–16.

Plutchik, R. 2001. The nature of emotions: Human emotions have deep evolutionary roots. *American Scientist*. **89**(4), pp.344–350.

Prechelt, L. 1998. Early Stopping - But When? , pp.55–69.

Qi, D., Tan, W., Yao, Q. and Liu, J. 2021. YOLO5Face: Why Reinventing a Face Detector.

Reinagel, P. and Zador, A.M. 1999. Natural scene statistics at the centre of gaze. *Network: Computation in Neural Systems*. **10**(4), pp.341–350.

Rensink, R.A. 2000. The dynamic representation of scenes. *Visual Cognition*. **7**(1–3), pp.17–42.

Ruder, S. 2016. An overview of gradient descent optimization algorithms.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C. and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*. **115**(3), pp.211–252.

Santurkar, S., Tsipras, D., Ilyas, A. and Madry, A. 2018. How does batch normalization help optimization? *In*: *Advances in Neural Information Processing Systems*., pp.2483–2493.

Saste, S.T. and Jagdale, S.M. 2017. Emotion recognition from speech using MFCC and DWT for security system. *undefined*. **2017-January**, pp.701–704.

Scherer, D., Müller, A. and Behnke, S. 2010. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. **6354 LNCS**(PART 3), pp.92–101.

Schlemper, J., Oktay, O., Schaap, M., Heinrich, M., Kainz, B., Glocker, B. and Rueckert, D. 2019. Attention gated networks: Learning to leverage salient regions in medical images. *Medical Image Analysis*. **53**, pp.197–207.

Scott Fortmann-Roe 2012. Understanding the Bias-Variance Tradeoff. [Accessed 7 February 2022]. Available from: http://scott.fortmann-roe.com/docs/BiasVariance.html.

Serrano, S. and Smith, N.A. 2020. Is attention interpretable? *In*: *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference* [Online]., pp.2931–2951. [Accessed 10 February 2022]. Available from: www.yelp.com/dataset_challenge.

Shan, C., Gong, S. and McOwan, P.W. 2009. Facial expression recognition based on Local Binary Patterns: A comprehensive study. *Image and Vision Computing*. **27**(6), pp.803–816.

Shimodaira, H. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*. **90**, pp.227–244.

Simonyan, K. and Zisserman, A. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Sitaula, C. and Hossain, M.B. 2021. Attention-based VGG-16 model for COVID-19 chest X-ray image classification. *Applied Intelligence (Dordrecht, Netherlands)*. **51**(5), p.2850.

Song, K., Yao, T., Ling, Q. and Mei, T. 2018. Boosting image sentiment analysis with visual attention. *Neurocomputing*. **312**, pp.218–228.

Sutskever, I., Martens, J., Dahl, G. and Hinton, G. 2013. On the importance of initialization and momentum in deep learning.

Viola, P. and Jones, M.J. 2004. Robust Real-Time Face Detection. *International Journal of Computer Vision*. **57**(2), pp.137–154.

Wang, C., Venkatesh, S.S. and Judd, J.S. 1995. Optimal stopping and effective machine complexity in learning *In*: *IEEE International Symposium on Information Theory - Proceedings*., p.169.

Wang, Q., Wu, B., Zhu, P., Li, P., Zuo, W. and Hu, Q. 2020. ECA-Net: Efficient channel attention for deep convolutional neural networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*., pp.11531–11539.

Wang, S.H., Zhou, Q., Yang, M. and Zhang, Y.D. 2021. ADVIAN: Alzheimer's Disease VGG-Inspired Attention Network Based on Convolutional Block Attention Module and Multiple Way Data Augmentation. *Frontiers in Aging Neuroscience*. **13**, p.687456.

Woo, S., Park, J., Lee, J.Y. and Kweon, I.S. 2018. CBAM: Convolutional block attention module *In*: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* [Online]. Springer Verlag, pp.3–19. [Accessed 14 April 2022]. Available from: https://link.springer.com/chapter/10.1007/978-3-030-01234-2_1.

Wu, C.H., Chuang, Z.J. and Lin, Y.C. 2006. Emotion recognition from text using semantic labels and separable mixture models. *ACM Transactions on Asian Language Information Processing*. **5**(2), pp.165–182.

Xu, K., Ba, J.L., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R.S. and Bengio, Y. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *32nd International Conference on Machine Learning, ICML 2015*. **3**, pp.2048–2057.

Yang, S., Luo, P., Loy, C.C. and Tang, X. 2015. WIDER FACE: A Face Detection Benchmark. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. **2016-December**, pp.5525–5533.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard and Lawrence D Jackel 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*. **1**(4), pp.541–551.

Zagoruyko, S. and Komodakis, N. 2016. Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention

Transfer. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*.

Zhang, P., Xue, J., Lan, C., Zeng, W., Gao, Z. and Zheng, N. 2018. Adding Attentiveness to the Neurons in Recurrent Neural Networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. **11213 LNCS**, pp.136–152.

Zhang, Q., Han, Z., Yang, F., Zhang, Y., Liu, Z., Yang, M. and Zhou, L. 2020. This paper is included in the Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation Open access to the Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation is sponsored by USENIX Retiari. *Osdi*., pp.919–936.

Zhao, G. and Pietikäinen, M. 2007. Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **29**(6), pp.915–928.

Zhao, Q. and Koch, C. 2013. Learning saliency-based visual attention: A review. *Signal Processing*. **93**(6), pp.1401–1407.

Zhi, R., Flierl, M., Ruan, Q. and Kleijn, W.B. 2011. Graph-preserving sparse nonnegative matrix factorization with application to facial expression recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*. **41**(1), pp.38–52.

Zhong, L., Liu, Q., Yang, P., Liu, B., Huang, J. and Metaxas, D.N. 2012. Learning active facial patches for expression analysis *In*: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* [Online]., pp.2562–2569. [Accessed 19 April 2022]. Available from: https://www.researchwithrutgers.com/en/publications/learning-active-facial-patches-for-expression-analysis.

Волянський, І. 2021. A DEEP DIVE INTO THE BASICS OF DEEP LEARNING. *Proceeding of the Shevchenko Scientific Society. Medical Sciences*. **65**(2).

# Appendix A
# Self-appraisal

## A.1 Critical self-evaluation

The project's goal was to develop a deep neural network that can successfully classify seven basic human emotions. In order to achieve good classification accuracies, the project aimed to utilise attention which is a novel concept in the field of computer vision. Overall the project achieved all the set-out tasks while also discovering improvements to established models for the emotion recognition task. For example, improvements were found for the existing ResNet and VGGNet architectures via a new activation function, ELU.

The paper also provided a range of discoveries, such as ResNetV2 performing significantly better on smaller datasets than the original ResNet and how significant attention is to improving models' performance. The extensive research also provides optimal parameters for the datasets and attention modules while providing the results to back up the claims.

However, the project had many constraints because it was trained and implemented on a local machine. More accurate performance metrics could have been achieved through a combination of 10-fold validation, which was impossible due to the computational requirement. Moreover, the FER2013 is the best dataset to compare metrics with due to its size and difficulty, but extensive tests could not have been executed.

In conclusion, the project successfully achieved the initially set goals by exploring attention modules and achieving 64% on FER2013, comparable to the human performance of 65.5% on this dataset. However, the best-performing models still have not achieved state-of-the-art performance compared to other projects in the field, which was a disappointing outcome.

## A.2 Personal reflection and lessons learned

Overall, the project was successful, and I'm proud of the final results. I have always had a keen interest in machine learning, but this project allowed me to research it in-depth and deploy a successful model while utilising the most recent discoveries in the field. After completing the project, I believe I gained a good understanding of machine learning in computer vision and can now apply such skills in future projects. Through the development of the project, I learned serval lessons.

Firstly, while researching new concepts, it was crucial to utilise multiple sources to understand them fully. This was particularly important while implementing the attention modules as the original research only publicised the code in PyTorch and not TensorFlow. This follows another lesson learned which was the ability to spot unexpected results, as errors in machine learning are not caused by syntax errors but by logical errors. To spot such errors, it is vital to have an expected outcome for each execution, such as the addition of the attention module should increase the accuracy. In runs where the expected outcome was not met, the ability to understand why is crucial in solving the problem.

Secondly, I found that implementation decisions may work for one image classification task but may not work for another for various reasons. Therefore, it is crucial not to generalize other projects' findings to your own.

Lastly, the project has taught me critical time and project management skills. More importantly, the project improved my ability to break down the project into individual tasks. I have learned to set a hard deadline for each task, ensuring that the project was completed in the desired time frame. Even if the deadline seemed far away, strict, short-term deadlines for each task made it easy to stay on track and ensured the project was completed to a high standard.

## A.3 Legal, social, ethical, and professional issues

### A.3.1 Legal issues

The BCS's code of conduct (BCS, 2020)addresses the legal ethics section as “having regard for the legitimate right of Third Parties”. In other words, the project must address intellectual property and data protection laws. The project utilized public implementations of pre-existing solutions for the CNN architectures, and the attention modules methods section was referenced in the writing and the code. The authors of VGGnet, ResNet, SEN-Net, ECA-Net, and CBAM published papers with clear implementation details of each architecture and made their code open-source for the public to use as long as it was referenced. Additionally, the code used was re-implemented in TensorFlow to not infringe on plagiarism. The facial detection software used for the project was also open-source and free to use if it was cited. The implementation of the application was referenced in the code and the paper.

Secondly, data protection laws were considered when dealing with databases. To acquire the needed data for CK+ and JAFFE, permission was required from the respective authors with an explicit declaration that the data would not be re-distributed. To adhere to these rules, the code published to GitHub is published without the specified databases to avoid re-distribution, and the use of the datasets is referenced in the paper. FER2013 dataset being part of global competition, is protected by the Database Contents License (DbCL) v1.0, which grants worldwide, royalty-free, irrevocable copyright access to the data. However, the dataset was still referenced in work to credit the creators of the dataset.

### A.3.2 Social issues

Social issues are problems that affect a larger number of people. Some of the most significant social issues stem from discrimination towards a group of people. When related to machine learning, the most common social issue is found within datasets that do not fully represent the entire population. It is vital to determine who is included and who is not when using any dataset. A large dataset, FER2013, was utilized to combat this issue, which is diverse in its samples. It contains different age groups, genders, and ethnicities hence it was the key metric to determine if the performance of the FER application is acceptable.

### A.3.3 Ethical issues

The general ethical concerts aim to avoid harm to other people, respect the right that others have to privacy and uphold the confidentiality of the subjects.

One of the essential concerns that apply to FER software is that machines cannot infer one’s actual emotional state. Therefore, FER systems should not claim to be able to achieve this simply from facial expressions. As stated before, emotion detection is a complex task, and the primary goal of FER systems is to classify emotions as if another human were to do so. In complex cases, people cannot fully determine each other’s emotions or even fully determine their emotional state. In other words, FER applications capture what the subject is trying to convey or what is perceived by the viewer. Even then, due to the complexity of human expression, they can be inaccurate. Not being explicit about this fact can cause negative ramifications if the application developers make claims about the users' mental state (Mohammad, 2021). This problem is closely related to the problem of physiognomy, which suggests we can determine the internal state from outwards appearance, which is not true and is actively harmful.

Additionally, these points follow the possible misuse of FER applications, for example, using it at airports to determine whether an individual is dangerous simply from their facial expressions. The ethical implication of FER software must be analysed for the use cases they plan to be deployed.

Lastly, as stated before, the CK+ and JAFFA datasets stated to not re-distribute the data to respect the subjects' privacy. FER2013 Dataset was built by gathering thousands of google images and was therefore made public. However, the authors did not clarify if each image sample followed ethical privacy concerns. Even though the authors provide everyone with the right to do anything with the data, the project will not re-distribute the data for ethical reasons.

### A.3.4 Professional issues

The project followed the professional guidelines specified by the British Computer Society (BCS) (BCS, 2020). Their rule of conduct states that the project should follow the ethics described above. The author should only undertake work within their professional competence, and the author should not claim any level of competence that they do not possess. Ensuring professionalism was upheld, much focus was placed on the research before any parts of the code were completed. Each section in the project contains references justifying the decisions made via credible sources, and any work that the author did not complete has been clearly outlined and referenced. Additional, feedback provided by the supervisor and assessor of the project was considered to develop better quality software and ensure that professional competence was upheld.

# Appendix B

## External Materials

External Code Repositories used in the code:

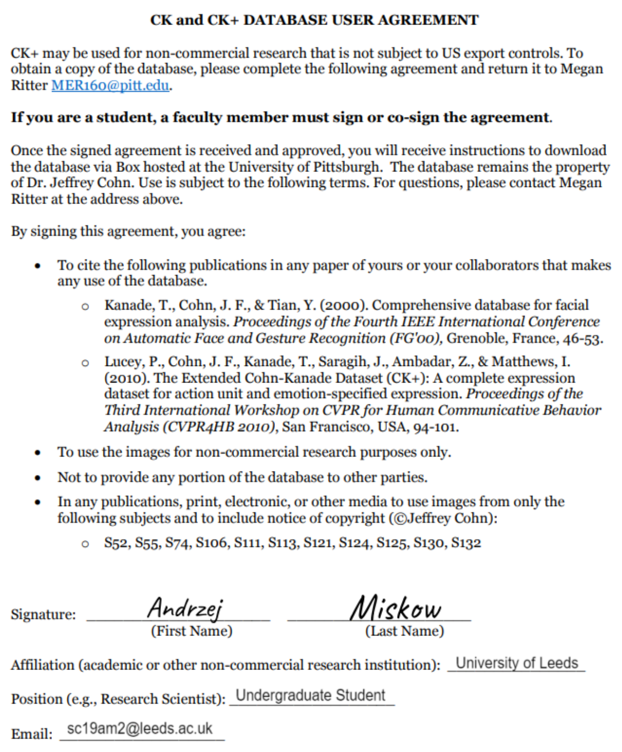
- YOLO5Face: Why Reinventing a Face Detector - (Delong Qi and Weijun Tan et al., 2021)

Implementation of CNNs was checked against their implementations in TensorFlow:

- Keras ResNetV1 - (Chollet et al., 2015)
- Keras ResNetV2 - (Chollet et al., 2015)

Datasets:

- CK+ (Figure B.1)
- JAFFE (Figure B.2)
- FER2013 (Manas Sambare, 2013)



**CK and CK+ DATABASE USER AGREEMENT**

CK+ may be used for non-commercial research that is not subject to US export controls. To obtain a copy of the database, please complete the following agreement and return it to Megan Ritter MER160@pitt.edu.

**If you are a student, a faculty member must sign or co-sign the agreement.**

Once the signed agreement is received and approved, you will receive instructions to download the database via Box hosted at the University of Pittsburgh. The database remains the property of Dr. Jeffrey Cohn. Use is subject to the following terms. For questions, please contact Megan Ritter at the address above.

By signing this agreement, you agree:

- To cite the following publications in any paper of yours or your collaborators that makes any use of the database.
  - Kanade, T., Cohn, J. F., & Tian, Y. (2000). Comprehensive database for facial expression analysis. *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG'00)*, Grenoble, France, 46-53.
  - Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., & Matthews, I. (2010). The Extended Cohn-Kanade Dataset (CK+): A complete expression dataset for action unit and emotion-specified expression. *Proceedings of the Third International Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB 2010)*, San Francisco, USA, 94-101.
- To use the images for non-commercial research purposes only.
- Not to provide any portion of the database to other parties.
- In any publications, print, electronic, or other media to use images from only the following subjects and to include notice of copyright (©Jeffrey Cohn):
  - S52, S55, S74, S106, S111, S113, S121, S124, S125, S130, S132

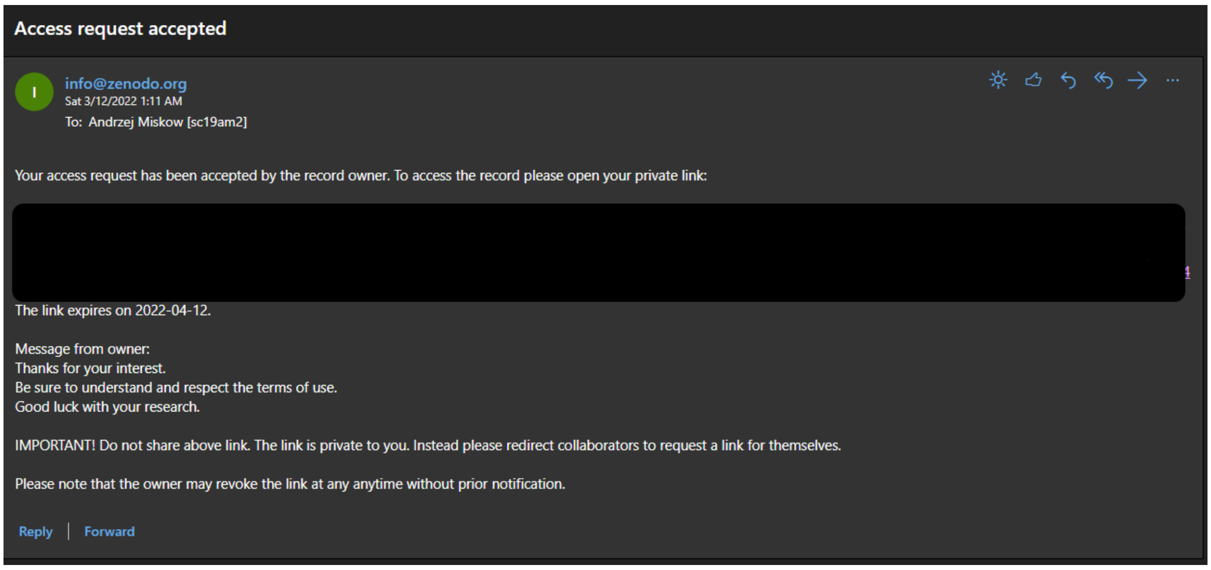
Signature: Andrzej (First Name) Miskow (Last Name)

Affiliation (academic or other non-commercial research institution): University of Leeds

Position (e.g., Research Scientist): Undergraduate Student

Email: sc19am2@leeds.ac.uk

**Figure B.1 : CK+ Request Form**



Access request accepted

info@zenodo.org
Sat 3/12/2022 1:11 AM
To: Andrzej Miskow [sc19am2]

Your access request has been accepted by the record owner. To access the record please open your private link:

[redacted]

The link expires on 2022-04-12.

Message from owner:
Thanks for your interest.
Be sure to understand and respect the terms of use.
Good luck with your research.

IMPORTANT! Do not share above link. The link is private to you. Instead please redirect collaborators to request a link for themselves.

Please note that the owner may revoke the link at any anytime without prior notification.

**Figure B.2: JAFFE Request Confirmation**

# Appendix C
# Additional Materials

## C.1 Research

### C.1.1 Representational Learning

Representational learning is a method that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification (Lecun et al., 2015). There are many learning representations however, in deep learning methods, they are formed by the composition of multiple non-linear transformations, intending to yield more abstract and more useful representations (Bengio et al., 2012). This learning method is used to create a network that consists of multiple levels of representation, with each layer having a different representation of the raw data. With enough layers and representations, complex features can be learned. The key aspect of deep learning is that humans do not design these levels of representation. Instead, the network learns them.

### C1.2 Convolution

In mathematics, convolution is an operation on two functions that produces a third function that represents how the shape is modified by the other. The convolution operation is typically denoted with an asterisk:

$$s(t) = (x * w)(t)$$

In CNN, the first function($x$) on which the convolution is performed is the input, the second function ($w$) is the kernel, and the output is the feature map. Convolutions are executed by moving the kernel over the input space, and at every location, matrix multiplication is performed to output the feature map.

### C.1.3 Motivation behind Convolution

Convolution use two important concepts that help improve a machine learning system: sparse interactions and parameter sharing (Lecun et al., 2015).

#### Spares interactions

In a traditional neural network, layers are fully connected. This means that every neuron in the input interacts with every neuron in the next layer. However, CNN's have spares interactions since the filter is smaller than the input. Because of this, an input matrix could consist of millions of units, but the filter will only interact with a small section of that input matrix, reducing the number of parameters in the network. Thanks to this property, neural networks need to store and compute fewer parameters, improving memory requirements and statistical efficiency.

**Parameter/Weight Sharing**

This property refers to using the same parameter for more than one function in the model. In standard neural networks, each parameter is used precisely once when computing the output of a layer. In CNN, each filter is applied to every location of the input. Hence through parameter sharing, rather than learning a separate set of parameters for every location, we learn only one set and share this result with other functions that are “tied” together. Parameter sharing does not reduce the computation time for forwarding propagation, but it further reduces the memory requirement for the model as fewer parameters need to be stored (Lecun et al., 2015).

### C.1.4 Spatial invariance

Spatial invariance means that the feature map is approximately invariant to small translations of the input. To elaborate, invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change. This property is useful if we care about whether a feature is present rather than where it is. For example, we need to know that the eyes and mouth are present rather than their exact location in emotion recognition. Additionally, if we pool over the outputs of separately parametrized convolutions, the features can learn which transformations to become invariant to. Hence, the network can learn to be invariant to the transformation of the input, as seen in figure 1.

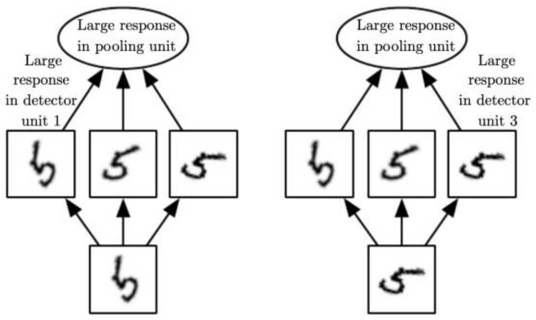

Large response in pooling unit
Large response in detector unit 1
Large response in pooling unit
Large response in detector unit 3

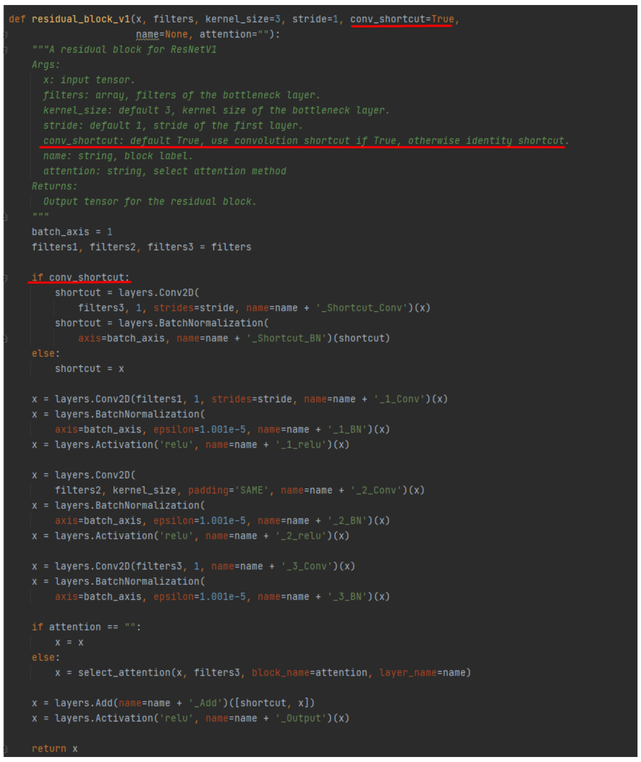
**Figure 1.4: Example of spatial invariance (Lecun et al., 2015)**

## C.2. Methods

### C.2.1 Splitting The Dataset

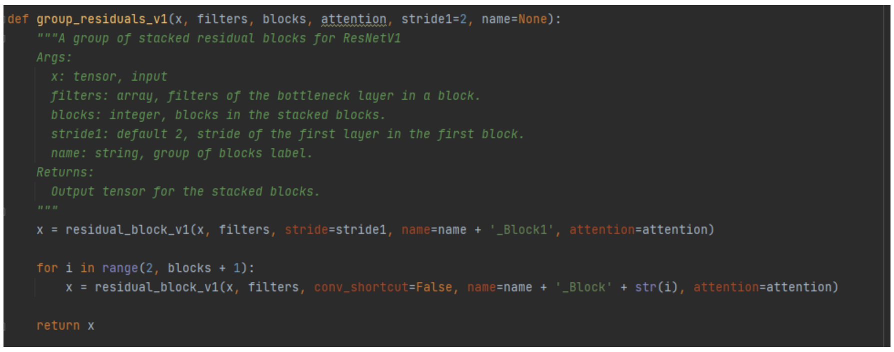
The dataset is typically split into three sets in machine learning: training, validation, and test set. The training set is used to fit the model, the validation set is used to evaluate the model during the training process, and the test set is used to evaluate the fully trained model on unseen data. Unfortunately, there is no golden ratio for how the three-set should split the original dataset. The research “An Evaluation of Training Size Impact on Validation Accuracy for Optimized Convolutional Neural Networks” (Barry-Straume et al., 2018) has shown that the size of the training set largely depends on the dataset size and the quality of the data. They have shown that good accuracy can be achieved on high-quality data even if only 40% of the data is used for training. Through experimenting with different dataset splits, the split which achieved the highest accuracy was: 80% training and 20% test split, with a further 30% of the training set being used for validation. Additionally, Increasing the training split past the 80% mark displayed poor final evaluation results on CK+ with some classes only including 1-2 samples.

### C.2.2 ResNet Implementation



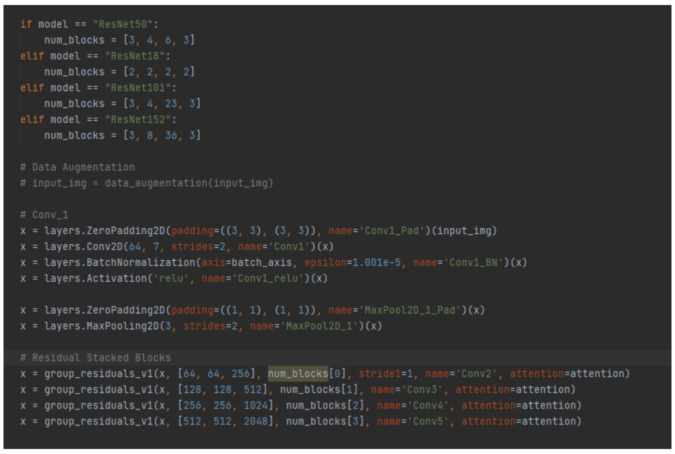
```python
def residual_block_v1(x, filters, kernel_size=3, stride=1, conv_shortcut=True,
                      name=None, attention=""):
    """A residual block for ResNetV1
    Args:
      x: input tensor.
      filters: array, filters of the bottleneck layer.
      kernel_size: default 3, kernel size of the bottleneck layer.
      stride: default 1, stride of the first layer.
      conv_shortcut: default True, use convolution shortcut if True, otherwise identity shortcut.
      name: string, block label.
      attention: string, select attention method
    Returns:
      Output tensor for the residual block.
    """
    batch_axis = 1
    filters1, filters2, filters3 = filters

    if conv_shortcut:
        shortcut = layers.Conv2D(
            filters3, 1, strides=stride, name=name + '_Shortcut_Conv')(x)
        shortcut = layers.BatchNormalization(
            axis=batch_axis, name=name + '_Shortcut_BN')(shortcut)
    else:
        shortcut = x

    x = layers.Conv2D(filters1, 1, strides=stride, name=name + '_1_Conv')(x)
    x = layers.BatchNormalization(
        axis=batch_axis, epsilon=1.001e-5, name=name + '_1_BN')(x)
    x = layers.Activation('relu', name=name + '_1_relu')(x)

    x = layers.Conv2D(
        filters2, kernel_size, padding='SAME', name=name + '_2_Conv')(x)
    x = layers.BatchNormalization(
        axis=batch_axis, epsilon=1.001e-5, name=name + '_2_BN')(x)
    x = layers.Activation('relu', name=name + '_2_relu')(x)

    x = layers.Conv2D(filters3, 1, name=name + '_3_Conv')(x)
    x = layers.BatchNormalization(
        axis=batch_axis, epsilon=1.001e-5, name=name + '_3_BN')(x)

    if attention == "":
        x = x
    else:
        x = select_attention(x, filters3, block_name=attention, layer_name=name)

    x = layers.Add(name=name + '_Add')([shortcut, x])
    x = layers.Activation('relu', name=name + '_Output')(x)

    return x
```

**Figure 2.1 Identity and Projection Blocks**



```python
def group_residuals_v1(x, filters, blocks, attention, stride1=2, name=None):
    """A group of stacked residual blocks for ResNetV1
    Args:
      x: tensor, input
      filters: array, filters of the bottleneck layer in a block.
      blocks: integer, blocks in the stacked blocks.
      stride1: default 2, stride of the first layer in the first block.
      name: string, group of blocks label.
    Returns:
      Output tensor for the stacked blocks.
    """
    x = residual_block_v1(x, filters, stride=stride1, name=name + '_Block1', attention=attention)

    for i in range(2, blocks + 1):
        x = residual_block_v1(x, filters, conv_shortcut=False, name=name + '_Block' + str(i), attention=attention)

    return x
```

**Figure 2.2 Group of Residuals Blocks**



```python
if model == "ResNet50":
    num_blocks = [3, 4, 6, 3]
elif model == "ResNet18":
    num_blocks = [2, 2, 2, 2]
elif model == "ResNet101":
    num_blocks = [3, 4, 23, 3]
elif model == "ResNet152":
    num_blocks = [3, 8, 36, 3]

# Data Augmentation
# input_img = data_augmentation(input_img)

# Conv_1
x = layers.ZeroPadding2D(padding=((3, 3), (3, 3)), name='Conv1_Pad')(input_img)
x = layers.Conv2D(64, 7, strides=2, name='Conv1')(x)
x = layers.BatchNormalization(axis=batch_axis, epsilon=1.001e-5, name='Conv1_BN')(x)
x = layers.Activation('relu', name='Conv1_relu')(x)

x = layers.ZeroPadding2D(padding=((1, 1), (1, 1)), name='MaxPool2D_1_Pad')(x)
x = layers.MaxPooling2D(3, strides=2, name='MaxPool2D_1')(x)

# Residual Stacked Blocks
x = group_residuals_v1(x, [64, 64, 256], num_blocks[0], stride1=1, name='Conv2', attention=attention)
x = group_residuals_v1(x, [128, 128, 512], num_blocks[1], name='Conv3', attention=attention)
x = group_residuals_v1(x, [256, 256, 1024], num_blocks[2], name='Conv4', attention=attention)
x = group_residuals_v1(x, [512, 512, 2048], num_blocks[3], name='Conv5', attention=attention)
```

**C2.2.3 Main ResNet Functon**

## C.3 Results

### C.3.1 Activation Functions

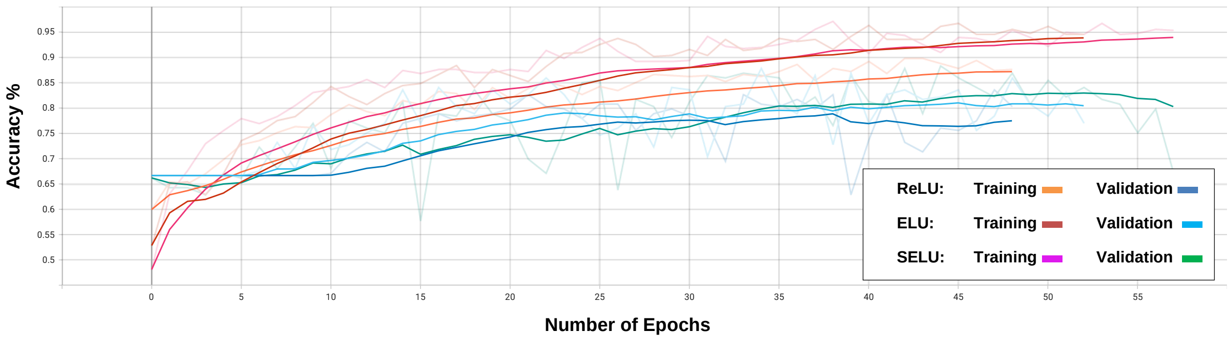


**Figure 3.1.1: Accuracy curves for activation function on the CK+ dataset**

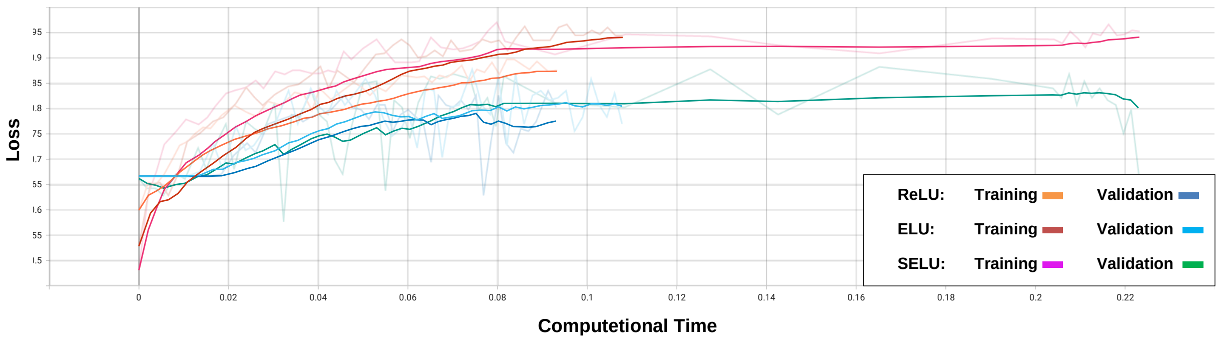


**Figure 3.1.2: Accuracy curves over time for activation function one the CK+ dataset**

| Activation Function | Accuracy |
|---|---|
| ReLU | 85.16% |
| ELU | 87.91% |
| **SELU** | **88.21%** |

**Figure 3.1.3: Performance of activation function one the CK+ dataset with ResNet-50**

SELU activation function achieved the best accuracy however it also doubled the execution time of the training. For this reason, ELU was chosen as the activation function as it still greatly improved the accuracy of the original ReLU without compromising execution time.

### C.3.2 Pooling

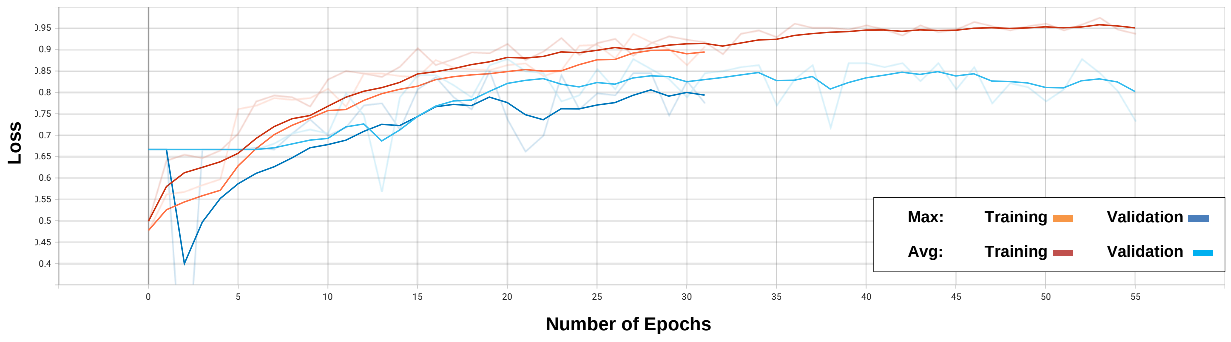


**Figure 3.2.1: Accuracy curves for pooling operation chosen in the final layer on the CK+ dataset**

| Pooling | Accuracy |
|---|---|
| Global Max Pooling | 82.97% |
| **Global Average pooling** | **88.46%** |

**Figure 3.2.1: Performance of for pooling operation chosen in the final layer on the CK+ dataset**

### C.3.3 CK+ Batch Size

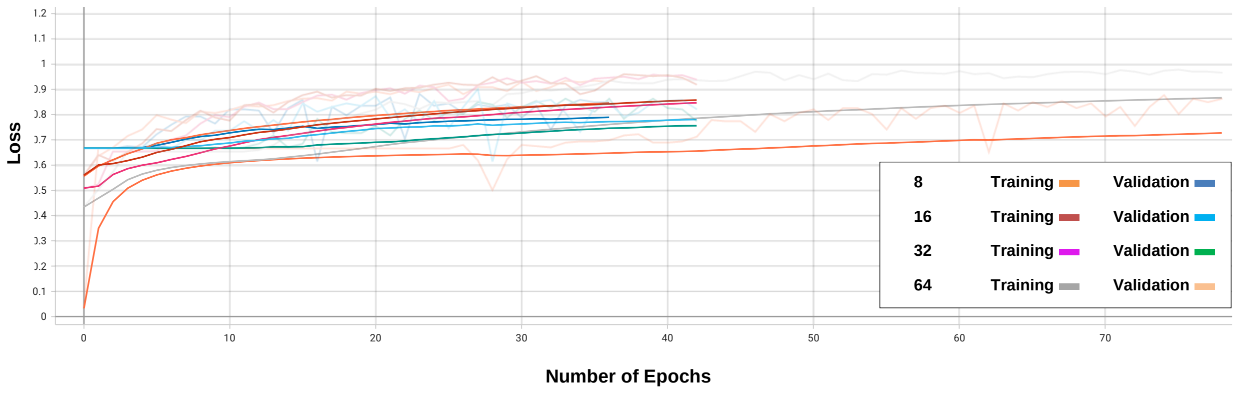


**Figure 3.3.1: Accuracy curves for different batch sizes on the CK+ dataset**

| Batch Size | Accuracy |
|---|---|
| 8 | **88.01%** |
| 16 | **88.01%** |
| 32 | 86.91% |
| 64 | 85.81% |

**Figure 3.3.2: Performance of different batch sizes on the CK+ dataset**

Batch sizes 8 and 16 achieved similar accuracies, with a batch of 8 slightly outperforming the 16 batch however the slight improvement in accuracy was not worth the computational time, which is doubled with the batch size of 8 competed to the batch size of 16.

## C3.4 JAFFE Batch Size

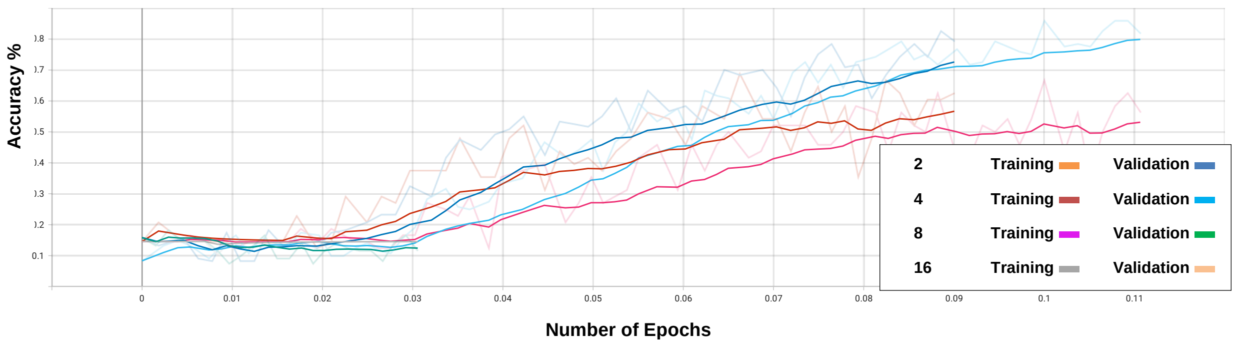


**Figure 3.4.1: Accuracy curves for different batch sizes on the JAFFE dataset**

| Batch Size | Accuracy |
|---|---|
| 2 | 60.00% |
| **4** | **68.89%** |
| 8 | 64.44% |
| 16 | 15.56% |

**Figure 3.3.2: Performance of different batch sizes on the JAFFE dataset**

## C3.5 FER2013 Batch Size

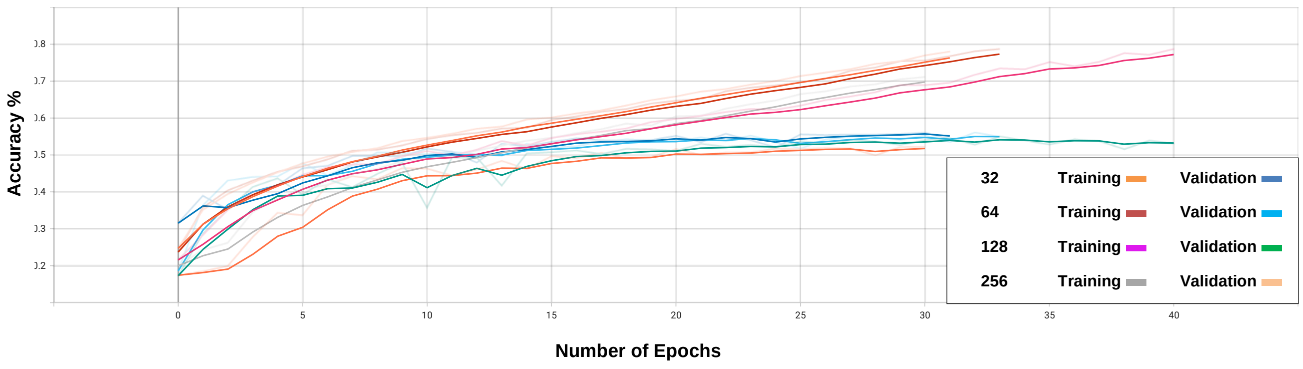


**Figure 3.5.1: Accuracy curves for different batch sizes on the FER2013 dataset**

| Batch Size | Accuracy |
|---|---|
| 32 | 57.02% |
| **64** | **58.05%** |
| 128 | 57.17% |
| 256 | 53.98% |

**Figure 3.5.2: Performance of different batch sizes on the FER2013 dataset**

## C3.6 SEN-Net Reduction Rate

### Training/Validation Accuracy over Epochs

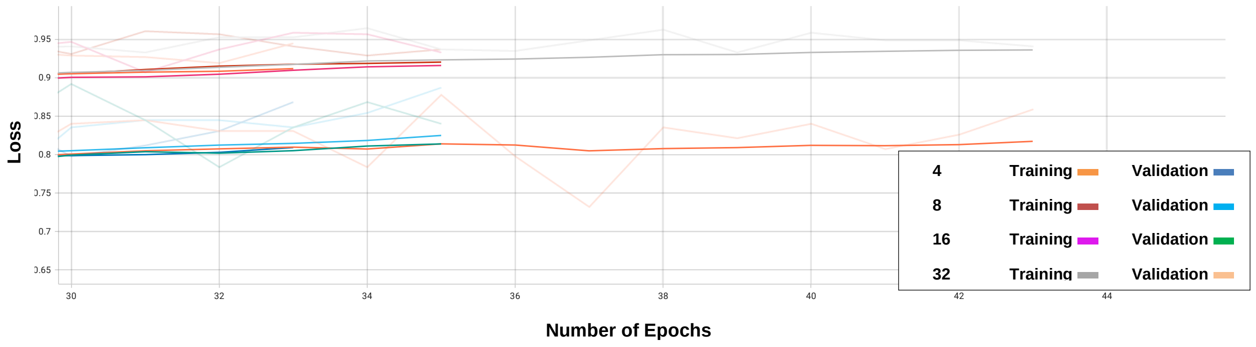


**Figure 3.6.1: Accuracy curves for different reduction ratios in SEN-Net on the CK+ dataset**

| Reduction Ratio | Number of Parameters | Accuracy |
|---|---|---|
| 4 | **33.56M** | 87.2637% |
| 8 | 28.53M | 88.4615% |
| 16 | 26.02M | **89.5604%** |
| 32 | 24.76M | 87.9121% |

**Figure 3.6.2: Performance of different reduction ratios on SEN-Net on the CK+ dataset**

The training/Validation graphs did not give too much information on which reduction rate was best. The 32 reduction rate achieved the best training accuracy but the lowest validation accuracy and the lowest evaluation accuracy. While on the other hand reduction size of 4 should have performed the best but in reality reduction size of 16 had the highest evaluation accuracy. This may be because of the data imbalance problem present in the CK+ dataset and its low sample size.

## C3.7 ECA-Net Kernel Size

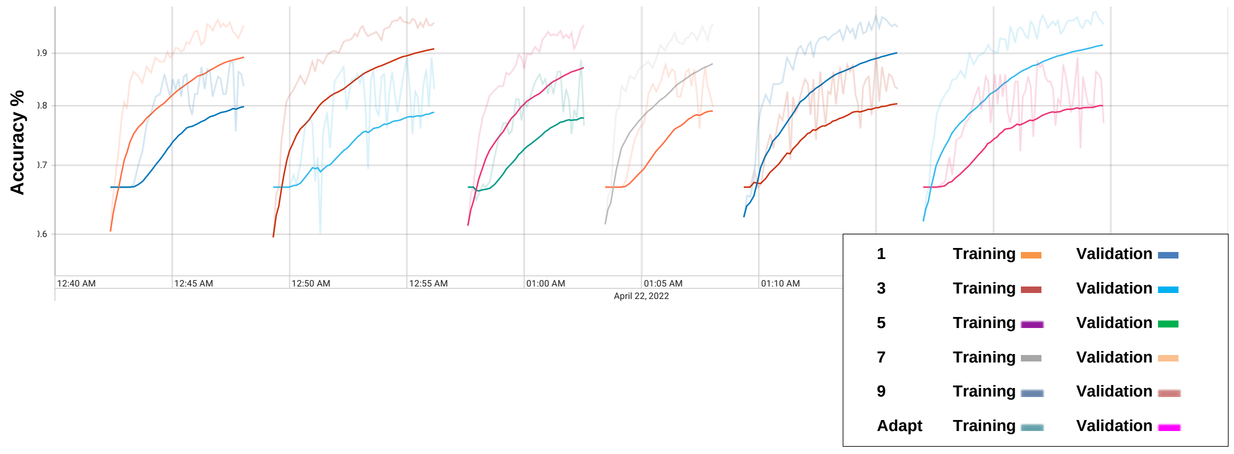


**Figure 3.7.1: Accuracy curves for different kernel sizes in ECA-Net on the CK+ dataset**

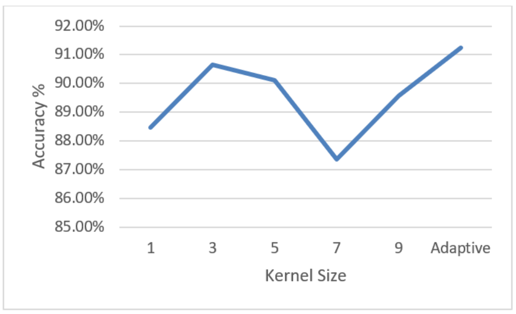


**Figure 3.7.2: Performance of different kernel sizes in ECA-Net on the CK+ dataset**

The results from TenserBoard are displayed on a wall graph as they were heavily overlapping, making the graphs hard to see. The graphs of training and validation follow the same pattern as the final results, with adaptive kernel size being the best option and kernel of size 3 being the best option for a numerical value.

## C3.8 CBAM Reduction Rate

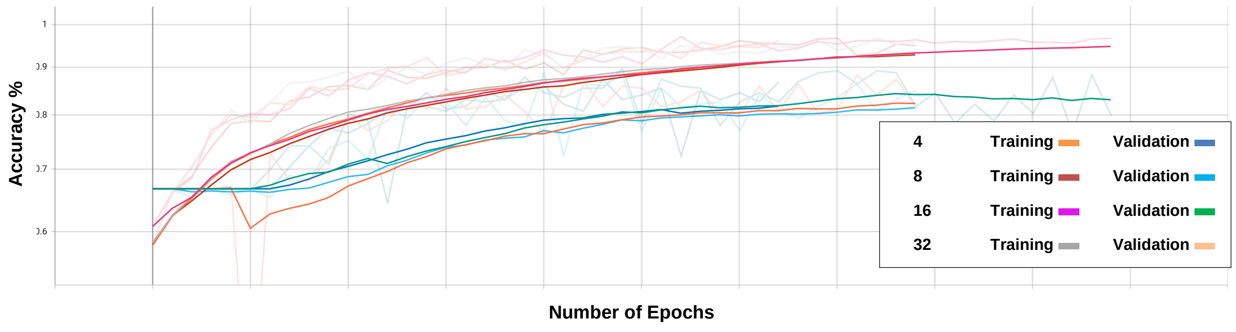


**Figure 3.8.1: Accuracy curves for different reduction ratios in CBAM on the CK+ dataset**

| Reduction Ratio | Number of Parameters | Accuracy |
|---|---|---|
| 4 | **33.57M** | 90.46% |
| 8 | 28.54M | 89.01% |
| **16** | 26.02M | **91.21%** |
| 32 | 24.77M | 90.66% |

**Figure 3.8.2: Performance of different reduction ratios in CBAM on the CK+ dataset**

## C.4 Final Results

### C.4.1 CNNs with Attetion

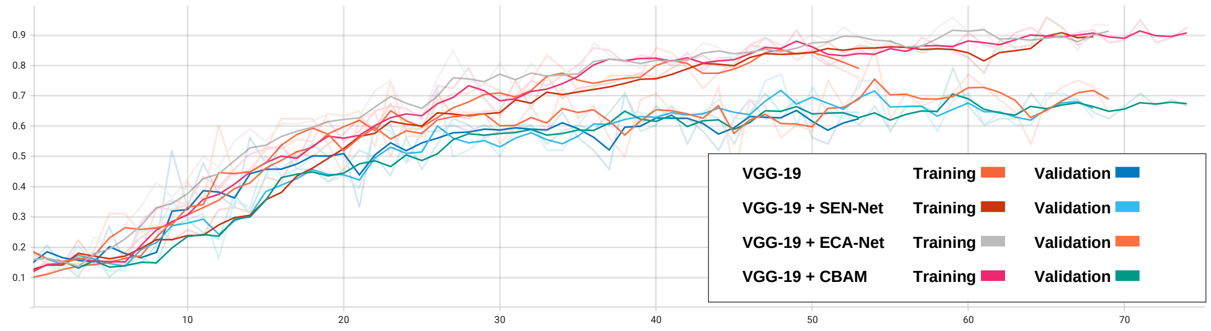


**Figure 4.1: Accuracy curves for VGG19 with attention on the CK+ dataset**

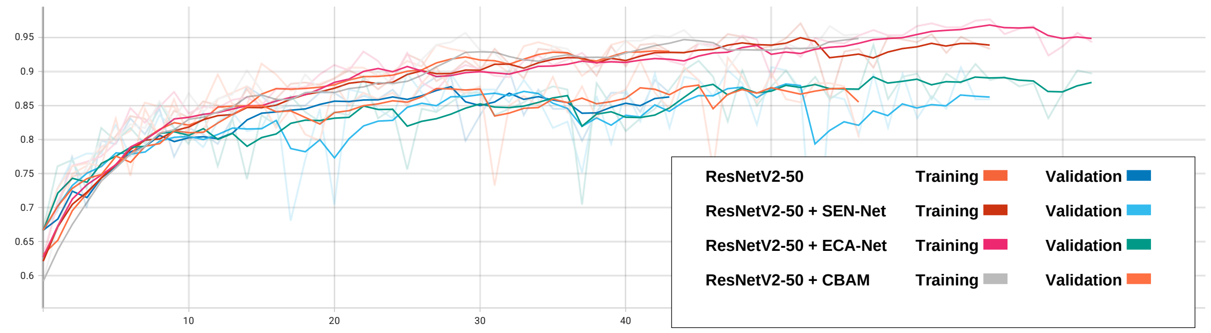


**Figure 4.2: Accuracy curves for ResNetV2-50 with attention on the JAFFE dataset**

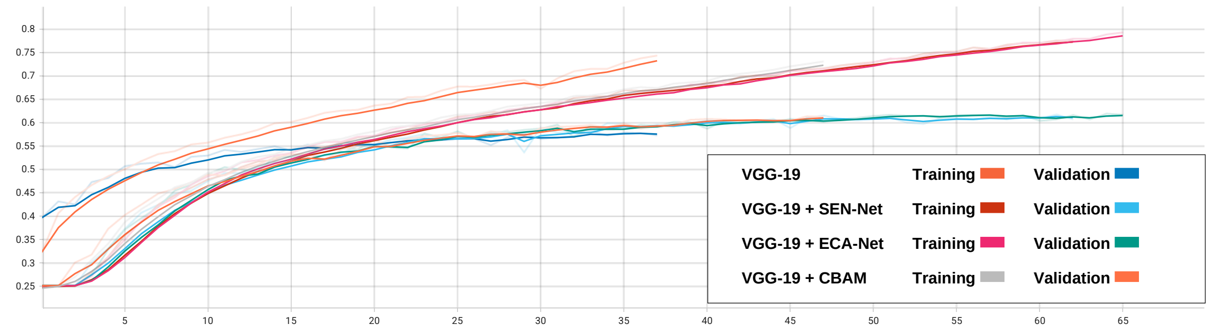
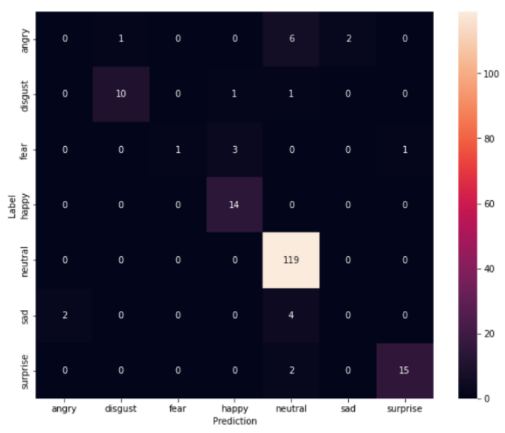
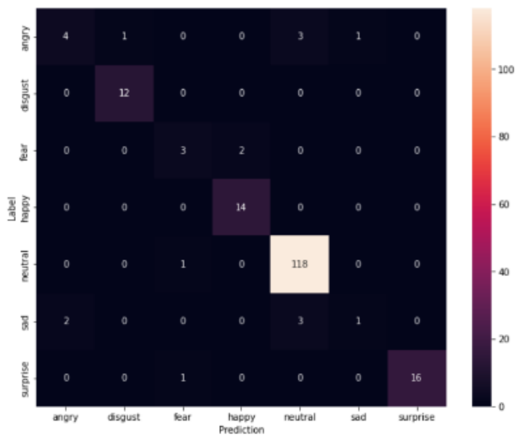


**Figure 4.3: Accuracy curves for VGG19 with attention on the FER2013 dataset**

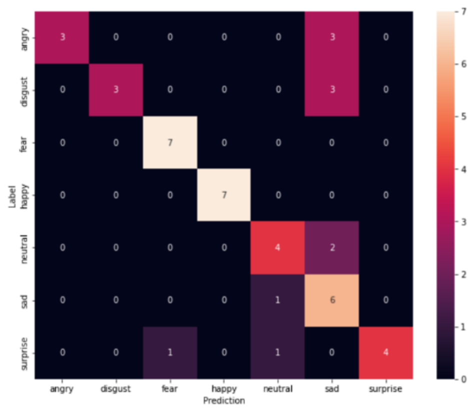


*a)VGG19*

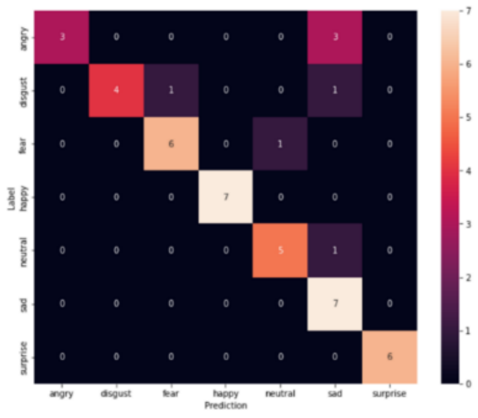


*b)VGG-19 + CBAM*

**Figure 4.4: Comparison of confusion matrices of the best performing models on the CK+**

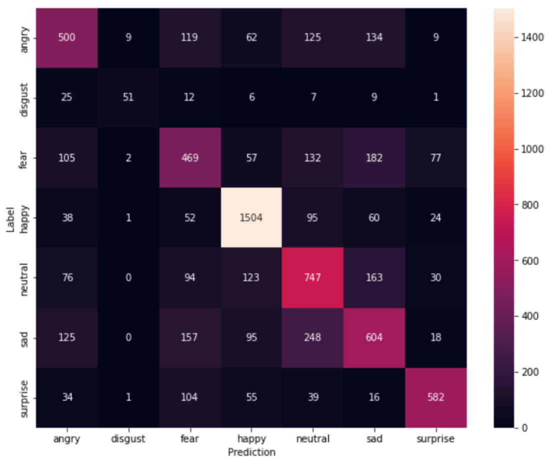


*a)ResNetV2-50*

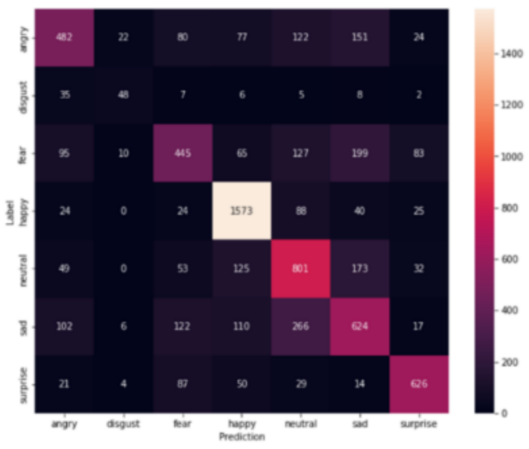


*b) ResNetV2-50+ CBAM*

**Figure 4.5: Comparison of confusion matrices of the best performing models on the JAFFE dataset**



*a)VGG19*



*b) VGG19+ CBAM*

**Figure 4.6: Comparison of confusion matrices of the best performing models on the FER2013 dataset**