Predicting Human Estimates of Image Similarity

Florica Margaritescu

MInf Project (Part 1) Report
Master of Informatics
School of Informatics
University of Edinburgh

2023

Abstract

Despite significant progress in machine learning model performance for vision tasks such as image classification and segmentation, accurately predicting human judgments of image similarity remains an open problem. In this study, we evaluate various machine learning architectures and training procedures on the task of predicting human estimates of image similarity using image triplets from three different datasets. Our contribution lies in examining the properties and correlations between model characteristics and resulting similarity prediction performance, as evaluated from different angles than previously explored by past studies. The aim is to provide additional insights into the differences between the machine learning model prediction process and human visual perception.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Florica Margaritescu*)

Acknowledgements

I would like to thank my supervisor Oisin Mac Aodha, who gave me valuable ideas throughout the entire process. Thank you very much for all of the feedback on my work and for the interest you showed in the project.

Table of Contents

Chapter 1

Introduction

This chapter aims to introduce the reader to the task that will be explored in this project, as well as explain the motivation and the goals behind the exploration of the task. The contributions resulted from the conducted experiments will also be summarised.

1.1 Task Description

The purpose of this project is to explore how well-performing a number of different machine learning models are at predicting human estimates of image similarity. Firstly, by human estimates of image similarity, we mean the judgment of a human in terms of which image from a set is more similar to a main image (an image that we compare all other set images to). Such a judgement can be collected by giving a survey participant 8 images and 1 main image [38, 1] and asking them to *"Rank the two most similar set images to the main image."* - illustrated in Figure 1.1. From this human similarity judgement, we extract image triplets which will make up our dataset. An image triplet will contain, for example: the main image (the query), the most similar image (reference 1) and the second most similar image (reference 2).

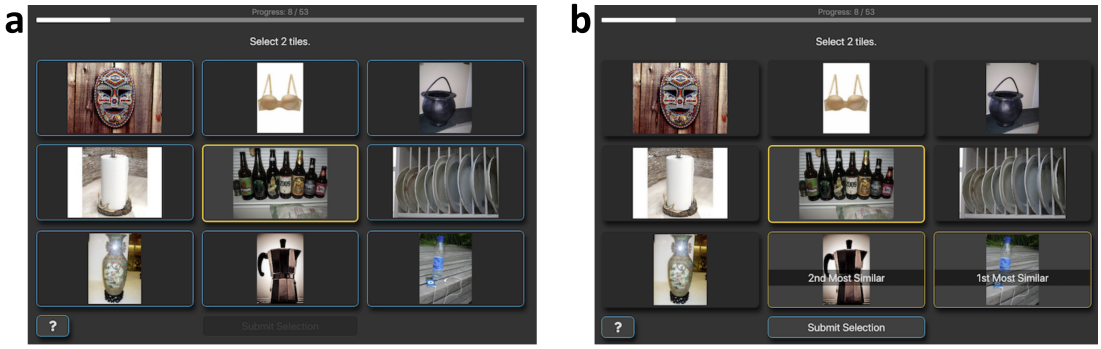

Figure 1.1: Example of the interface used in a human similarity judgement collection survey. (a) The participant is given 9 images, the highlighted middle image represents the main image, the rest are the image set. (b) The participant chooses (in order of their ranking) the most and second most similar image to the main image. Image and study credit: [38].

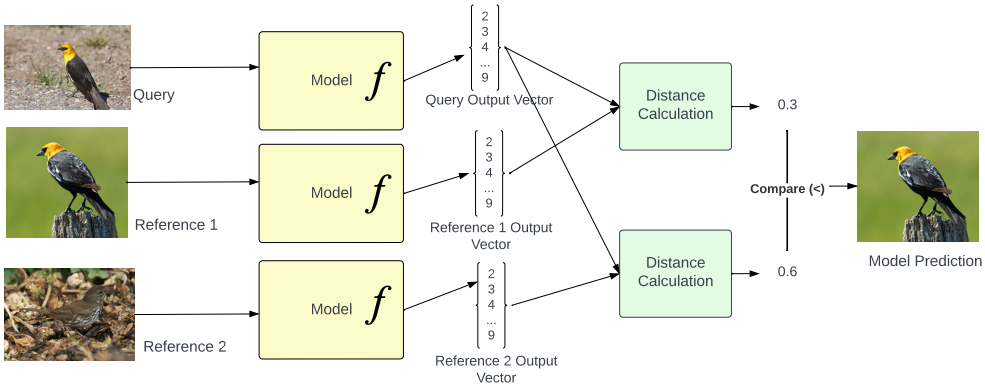

Figure 1.2: An illustration of how one would use a machine learning model to predict image triplet similarity originating from a dataset of human judgements.

Multiple trials alike the one illustrated in Figure 1.1 will be conducted, resulting in a single (for the purpose of this chapter) triplet each.

Now that we have a dataset of triplets, which represent human judgements of similarity, we shall use a machine learning model, which can be momentarily thought of as a mathematical function mapping an input to an output, to predict which reference is the most similar to the query. We essentially give the model the three images, and "ask" it to tell us which reference it considers to be most similar to the query. If the decision of the model is the same as that of the human - that reference 1 is more similar to the query - then the model made a correct prediction.

The model maps each input image in a triplet to a vector. To evaluate whether the model considers one reference image to be more similar to the query image, we simply calculate the distance of the output vectors of the query and each reference image. The closer in distance the vectors, the closer in similarity does the model consider the images. An illustration of this can be seen in Figure 1.2.

The purpose of this project is to evaluate a number of models in the manner exemplified above,as well as to train these models - by training, we mean allowing the model to "learn" the particularities of our dataset so it can make better predictions.

1.2 Motivation

The purpose of evaluating and training models on this task is striving to understand what it is that sets apart the model decision making process and human perception for the purpose of contributing to the efforts of aligning the model prediction process to human visual perception.

The follow-up question is: what are the benefits of aligning the model prediction process to human visual perception? We enumerate a number of desirable features of human visual perception that models do not innately present:

- Ability not to be affected by small perturbations to images: these perturbations are imperceptible to humans, however they drastically affecting model predictions [17]);
- Contextual understanding [38]: humans can choose similar images from the most unlikely pairs: an image of a cigarette and an image of bottle of beer can be considered similar – as both products present age restrictions;
- Humans are able to learn new concepts from very few/single examples - a phenomenon called one-shot learning [51], [34]. Models need to train on very large amounts of data in order to "learn" a task and thus make quality predictions.

Not only, past studies [14, 1] have also discussed how, by approaching machine decision making to human perception, we unexpectedly also improve model performance in tasks different to the one described in Section 1.1.

1.3 Contributions

The main contributions of this project are:

- An evaluation of a number of different machine learning model architecture performance on three human similarity judgement datasets [38, 1, 53];
- An evaluation of training methods for the task described in Section 1.1;
- An evaluation of correlations between machine learning model hyper-parameters and properties and their performance on the task described in Section 1.1;
- Exploration of other possible correlations between model performance and properties of the dataset that it is applied to.

The experiments explored in this project had the purpose of extending previous experiments conducted by other studies or offering a new angle of exploration.

Chapter 2

Background

The aim of this chapter is to give a high-level overview of essential background related to human vision and perception, machine learning models and human judgement datasets to be used in this project. This review shall be explained in enough detail (and more thoroughly explained in later chapters) to pave the way for the discussion concerning past studies revolving around the topic of machine learning models predicting human estimates of image similarity. The purpose of the chapter is to provide the necessary background information for obtaining a general understanding of the methods used throughout this project, as well as of the previous work concerning the topic and thus how this project fits in with past studies.

2.1 Human Vision and Perception

Human vision and human perception [10], related yet distinct concepts [27, 44], both play a role in assessing image similarity. Human vision processes the raw visual data of images (colours, textures, spatial information) and is thus concerned with the process of receiving and processing raw visual information through the visual system involving the eye and the brain. On the other hand, human perception refers to processes which integrate the information and interpretation gathered from multiple senses to obtain an all-encompassing understanding of the environment. When analysing an image, human perception considers more complex cognitive processes (memory, attention, contextual understanding), as well as visual information from human vision processes to obtain a subjective [43] image interpretation. Figure 2.1 illustrates the difference between vision and perception via a well-known illusion.

To exemplify this difference in the context of assessing image similarity, we may think of giving an individual two different images depicting the same scene, however, under different lighting conditions and asking if the two images are similar. When human vision processes the raw information, disparities between the visual information generated for the two images will arise: human vision alone may thus be unreliable in assessing similarity. Human perception applies higher cognitive processes, such as context knowledge and memory, to this raw information, to help us conclude that the images represent the same scene.

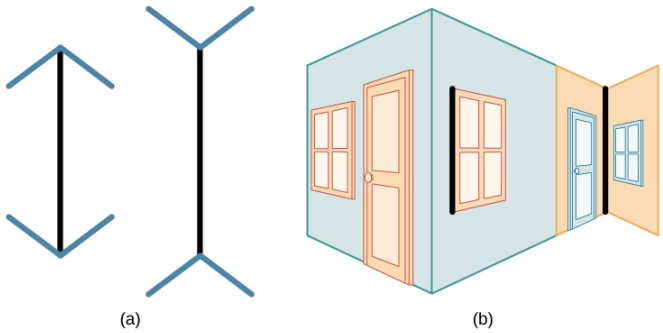

Figure 2.1: The Müller-Lyer illusion [32] depicts the difference between vision and perception. Although presented with two identical lines (vision confirms this), we assess them to be of different lengths (due to depth perception). (a) Arrows at the ends of lines may make the line on the right appear longer. (b) When applied to a three-dimensional image, the line on the right again may appear longer. Image credit: [44].

The purpose of highlighting the difference between vision and perception is due to the fact that although certain machine learning model architectures (namely Convolutional Neural Networks - Section 4.1) have been inspired by the human visual system [33], studies show that other architectures (Transformers - Section 4.1) may be in better alignment with human image perception [48, 12] - Figure 1 (a),(b) and (c).

2.1.1 Properties of Human Vision

Given two images and the task of analysing similarity, we may logically assume that humans assess similarity given factors such as color, shape, texture and overall image shape composition; as well as human subjectivity due to personal preferences and experiences. Additionally, us humans can also use context and current knowledge to decide upon image similarity; for example: we may deem a picture of a bottle of alcohol and a picture of a cigarette as similar due to our prior knowledge that both products are age-restricted [38]. Roles of elements [24] depicted in two images are also a factor - for example, we may deem similar a picture of a lion (predator) and that of an antelope (prey), despite not undertaking the same roles, they undertake roles within the same relation. Less obvious properties - relevant in later comparison to machine learning models - of human vision and perception are:

- The ”shape bias” [37], [29]: humans are biased towards shape when analysing an image (that is, we are likely to assign more importance to shapes than to colours or textures present in an image);
- Humans are strongly influenced by Gestalt shape when perceiving images - Gestalt principles ([1], [44] - Chapter 5.6) essentially describe how humans, when given a group of objects - in our case elements of an image - perceive the entirety of the picture before individual objects;
- Human similarity judgements are asymmetric [49], [1]: given images A and B, the human-perceived pair-wise similarity of A and B may not be equal to that of B and A;

- Humans vision and perception are robust [14]: we are able to generalise well to new data in the sense that given new data - an image for example, we are able to make correct inferences as to what the image represents. Additionally, humans are able to learn new concepts from very few/single examples - a phenomenon called one-shot learning [51], [34]. These two properties are especially desirable by machine learning models.

2.2 Datasets of Human Judgements

A number of past studies [38], [1], [53] infer qualities of human perception of image similarity from behavioural evidence drawn from tasks given to human participants where participants are given a number of images (including a query image - the main image that all other images are compared to) and are asked to select or rank these images with respect to how similar they are to the query image. Figure 1.1 presents the user interface used in such a judgement collection human study (Chapter 3 details more collection methods).

Following such a collection method, the resulting datapoint featuring in our dataset is represented as a triplet of images: containing a query image in relation to which participants were asked to rank other images, an image that human participants deemed to be the most similar to the query image and lastly, the third image may either be an image deemed as the second most similar to the query image by human participants, **or** an image deemed dissimilar to the query by human participants. Such a triplet example is present in the introductory illustration Figure 1.2.

As described in Section 1.1, the task that our project is concerned with is using machine learning models to predict human estimates of image similarity. To test machine learning models on this task, the following human similarity judgement datasets have been chosen based on the distinct properties that each presents: a general dataset spanning multiple categories of images [38], a bird species dataset spanning four umbrella species [1], and lastly, a distortion-based dataset [53].

2.3 Introduction to Neural Networks

This section aims to introduce neural networks (the machine learning models our project is focused on) at a high enough level for the reader to confidently proceed onto the discussion concerning past work. Chapter 4 goes into further depth explaining these models.

Neural networks are very loosely inspired by the structure of the human brain and are composed of a large number of interconnected nodes (neurons or units - a mathematical function) organised into layers. When given an input, each node on a layer passes its output to the nodes in the following layer. Outermost layers are called input and output layers, whilst every layer situated between these is called a hidden layer. This project concerns Deep Neural Networks (DNN) which consist of multiple hidden layers. We also see activation function layers (non-linear) between the aforementioned layers

(an activation function is thus applied to the output from a unit on a layer before passing it onto a unit on the next layer). These activation functions allow the network to approximate any [6] non-linear continuous function. Figure 4.1 (a) depicts an illustration of a simple network. A neural network is able to ”learn” the correct mapping between an input and an output by gradually adjusting the parameters making up the mathematical functions that it contains - this process is called training.

Deep neural networks present various different architectures where we see varied operations performed by nodes and different ways of interconnecting nodes and layers. The two main architectures that will be often encountered in the following discussion regarding past studies are Convolutional Neural Networks (CNN) and Transformers.

Convolutional Neural Networks or CNNs [16]) have been designed with the intention of processing data with a grid-like topology such as images and have been inspired by the biological visual system [33], [23]. Their name comes from the fact that they use matrix convolution operations.

Transformers [50] are neural networks which employ attention mechanisms to weigh the importance of different input components when predicting an output. Transformers are thus able to “focus” on different parts of the input at different times, all dependent on relevancy to task at hand. Transformers are fairly new, proposed in 2017 [50] and designed with sequences in mind (such as variable sequences of words), not for single datapoints (such as a single image) - however, they have been adapted to be used on images [8].

The below studies will also refer to ”features” and ”embeddings” (we will consider them interchangeable terms): neural networks can be used to extract features/embeddings from images (or other raw data). These features are matrices of values which makes sense to our network and which the network uses to represent essential information (we may think of such a feature as the presence or absence of an edge). The hidden layers extract these features and, as we go deeper into the network, each layer learns increasingly complex representations of the input. The output of the final hidden layer is used as a feature representation for the input image.

2.4 Related Work

The task of predicting human estimates of image similarity is still an open problem [30]. Performance on this task has not yet reached acceptable performance levels [38] when compared to other image-centred machine learning tasks: for example, CNN’s can achieve over 99% accuracy [31] on the simple MNIST dataset; on a more complicated dataset, such as ImageNet [41] (a very large dataset of images containing 14 million spanning more than 20,000 categories), a deep CNN such as ResNet-152 [20] is able achieve around 96% top 5 accuracy on image classification tasks[1].

The desirability of better predicting human estimates of image similarity is due to the assumption that by approaching machine learning model representations and human per-

[1] As indicated by the table of ImageNet pre-trained PyTorch models: https://pytorch.org/vision/stable/models.html

(a) Texture image
81.4% **Indian elephant**
10.3% indri
8.2% black swan

(b) Content image
71.1% **tabby cat**
17.3% grey fox
3.3% Siamese cat

(c) Texture-shape cue conflict
63.9% **Indian elephant**
26.4% indri
9.6% black swan

Figure 2.2: Classification prediction of a standard ResNet-50 (CNN) of (a) a texture image of elephant skin; (b) an image of a cat (with both shape and texture cues); and (c) an image with a texture-shape cue conflict (the texture belongs to an elephant, the shape to a cat). The CNN seems to classify the image according to its texture. Image and study credit [13]

ception may give models the same beneficial properties of human vision and perception as described in Section 1.2. However, the past few years have seen an improvement in the amount of progress towards understanding inner representations of machine learning models and how they compare to human perception; in improving machine learning predictions of human estimates of image similarity; as well as in compiling extensive datasets of human judgements to be used for model training and evaluation.

The reason for not yet being able to reach top performances in predicting human estimates of image similarity may be the nature of what we are predicting: we are not predicting/classifying a "universal truth", rather, we are predicting estimates – not all people agree on a similarity judgement, judgements are subjective to individual experience as discussed in Section 2.1, and the decision-making behind a similarity judgement may differ from person to person. The fact that this is a problem without a yet effective solution justifies the purpose of our study.

2.4.1 Aligning Models to Human Perception

Past studies, described below, have found various differences and similarities between human vision/perception (properties described in Section 2.1.1) and model internal representations (for various architectures) given the task of analysing images. The same studies also found that models which approach (or are modified as to approach) these human vision/perception properties, perform better on the task of predicting human estimates of image similarity.

- As mentioned in Section 2.1.1, humans are biased towards shape (that is, it is the shapes present in an image that contribute the most to our judgement). On the other hand, CNN models trained on ImageNet have been found [13] to be biased towards the texture of an image. This is illustrated in Figure 2.2 where we observe the network obtain conflicting cues and seemingly making a decision based on the texture cue. The same study [13] also shows how by training the same model on a stylized version of ImageNet, which aims to remove the texture

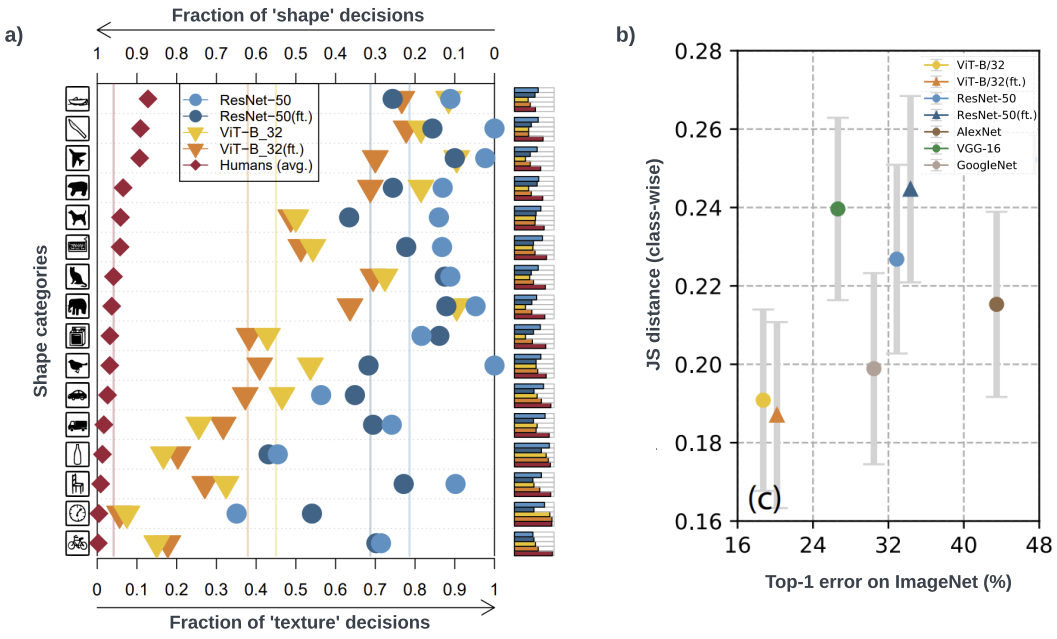

Figure 2.3: Image and study credit [48]: (a) Illustrating the contribution of shape and texture to a model's classification decision: "ft." refers to fine-tuned models - explained in Chapter 4. (b) Illustrates the error (the lower the better) of each architecture (note: all architectures besides ViT are CNN models) on ImageNet classification.

bias in favour of a shape bias, the model provides a much better fit for human behavioural performance and consequently (an unexpected beneficial outcome), yields improved object detection performance and robustness towards a wide range of image distortions.

- On the topic of shape and texture bias, another study [48] compares the shape bias of ViT (Visual Transformer) and CNN models, trained on ImageNet[2]. The study shows that the ViT architectures present a higher shape consideration (closer to humans) than CNN architectures and that they also outperform CNN architectures in ImageNet classification tasks - findings illustrated in Figure 2.3.

- As mentioned in Section 2.1.1, humans are strongly influenced by Gestalt shape when perceiving images - that is, we perceive the image in its entirety. Oppositely, a study proposed that CNN models [3] base predictions on local features. Not only, the study conducted experiments similar to those in Figure 2.2 by using the shape of an animal, yet the texture of an object, with the purpose of showing how that sways model prediction towards object texture (which it did).

- A study [1] focused strictly on obtaining higher accuracy of machine learning models on predicting human estimates of image similarity (on the bird species dataset detailed in Chapter 3) and succeeded in doing so by: linearly transforming, and namely reducing in size, of deep embeddings (further explained in Chapter 4

[2]Specifically on the ImageNet-21K and ILSVRC-2012 datasets [41].

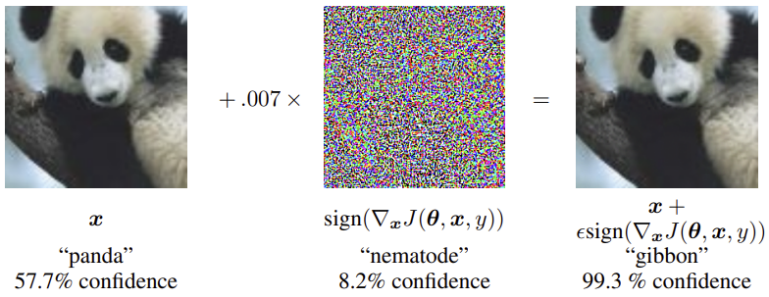

Figure 2.4: *x* is the original image - whilst the original image and the perturbed right hand-side image are identical to humans, they significantly shift model predictions. Image and study credit [17]

- essentially embeddings capture the most important information about an input) to better adhere to human representations raised accuracy of prediction of human binary choice on the bird species dataset from 72% to 89%. Not only, the study has also shown that allowing models to express asymmetric similarity (alike humans as discussed in Section 2.1.1) further improves explanatory power (a clearer and more interpretable model decision-making process).

- What humans would consider imperceptible perturbations to images (these intentional perturbations are called adversarial examples) has been found to thoroughly affect model image predictions (for classification tasks) [17]. Whilst humans do not perceive these perturbations at all (Figure A.1), specific models' (such as CNN GoogLeNet [46]) predictions change significantly (even reaching misclassification rates of 99% on simple datasets such as MNIST[3]).

- Other studies look into the error distribution difference for image classification tasks between models and humans [48, 11]. It has been shown that error distribution of CNN models is farther away from that of humans (far away enough to indicate very different strategies implemented by humans and CNN models in image classification decision-making [11]); whilst ViT's (Transformer) is closer.

2.4.2 Model Performance and Perception Score Correlation

Lastly, a very recent study [30] poses and answers the question of whether there is a possible correlation between the accuracy that a model managed to achieve on a vision task (such as image classification) and its ability to match human perception (also observed by [38]). Firstly, the aforementioned study uses perceptual scores (Learned Perceptual Image Patch Similarity (LPIPS))[53] which represents the distance between the feature representations of two images as generated by a DNN (in the paragraph below, the higher this score, the closer the representation is considered to be human perception). The models below are tested on the distortion BAPPS dataset [53] detailed in Chapter 3.

The study [30] indicated that there is an inverse-U relationship between model accu-

[3] A dataset of 60,000 small square grayscale images of handwritten digits ranging from 0 to 9, along with their corresponding labels.

racy on ImageNet classification accuracy and perceptual score on the BAPPS dataset: perceptual score improves as accuracy improves up to a certain point, afterwards it worsens. Not only, it was found that overparametrized models tend to perform worse than simpler models (which goes against expectations - provided that higher complexity generally seemed to result in higher accuracy on ImageNet classification tasks). This could indicate that a looser fit (and not overly-parametrized) model is a much better estimate for how humans perceive image similarity. Not only, modern and high accuracy yielding models (whether they are CNNs or Transformers) seem to in fact perform worse than their simpler counterparts (again going against expectations). Figure 2.5 illustrated this inverse-U relationship and how the best perceptual scores are obtained by architectures with moderate accuracy.

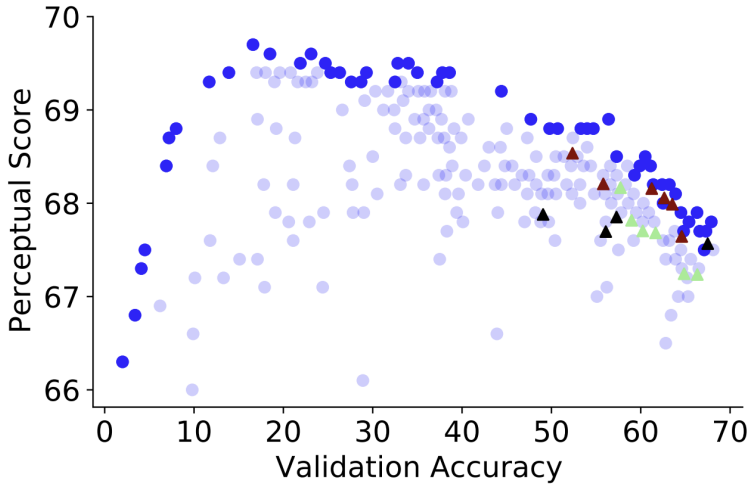

Figure 2.5: Each blue dot represents an ImageNet classifier, with EfficientNet [47] (CNN), ViT (Transformer) and ResNet (CNN) being marked as green, black and red triangles respectively. The lowest and highest Perceptual Score observed are 64.3 (using an untrained network) and 68.9 (AlexNet [28]) respectively. Image and study credit [30]

Chapter 3

Datasets

This chapter aims to offer a detailed discussion of the datasets used in this project and how these datasets were pre-processed and utilised by the machine learning models we will be evaluating. The chapter covers the collection, purpose and description of each dataset and shall also give the reader an idea of how ”difficult” of a challenge each dataset may pose and the value that each dataset will bring to our discussion when evaluating different models.

From this point onward, we will be using the terms **query** and **reference** image. The query image is the ”main” (point of comparison) image that the rest of reference images will be compared to.

3.1 Evaluation Datasets

3.1.1 ImageNet-HSJ

The Human Similarity Judgments extension to ImageNet (ImageNet-HSJ) [38] has the purpose of supplementing the ILSVRC (Large Scale Visual Recognition Challenge (ILSVRC) [41]) validation set with human similarity judgements.

The way that the data was collected is by using the images in the ILSVRC validation set (containing 50,000 images spanning 1,000 categories - some examples from the dataset are shown in Figure 3.1) and conducting surveys via Amazon Mechanical Turk (AMT)[1]. Each survey session consisted of 50 trials and during each trial, the human participant was shown nine images in a grid as shown in Figure 1.1. The center image being the query and the surrounding images the references. Participants selected the two reference images most similar to the query (the order of their selection indicated which reference they thought was most similar and second most similar to the query). We will refer to this collection mechanism as *8 rank 2* collection.

The images per trial were not selected in a random manner due to the high number of classes and the high probability that a trial will consist of images from highly dissimilar categories - which will only encourage participants to make random guesses with

[1]Platform for online task outsourcing: https://www.mturk.com

Figure 3.1: An example of images present in the ImageNet validation set. Image and study credit: [41].

regards to similarity (which are not intellectually valuable). To instead create trials that reveal consistent perceptual beliefs, an active learning paradigm was used ([38], Section 4). Moreover, there were four catch trials for every session (two in the first 20 trials and two in the last) to assess participation effort. For these trials one reference image was a mirror-image of the query image. For each session, catch trial performance determined the sample weight of the trials - in our project we only use images from trials where none of the catch trials were incorrectly answered.

As it will be described in Section 3.2, this dataset will be split into triplets. Figure 3.4 (c) and (d) offers an overview of how ImageNet-HSJ triplets appear. The reason for choosing this dataset was the variability of the categories that it spans (and the complexity that this brings) and the way in which it was collected (each trial having the potential to bring forward meaningful psychological behaviour).

3.1.2 Birds-16

The Birds-16 [1] dataset was collected as part of a study which attempted (and succeeded) to improve model performance at predicting human estimates of image similarity (binary choice prediction). The model was given three images, one query and two references, the task being to determine the reference most similar to the query.

Similarly to ImageNet-HSJ, the judgement collection was performed on AMT and used a previously collected dataset of bird images of similarity judgements [39] (208

Figure 3.2: Image and study credit: [39]. All images in the Birds-16 dataset where each column depicts a single species.

images in total, containing four bird families, 4 distinct species for each family and 13 specimens per species - significantly smaller than the ImageNet validation dataset). Figure 3.2 gives an overview of the birds dataset. Trials during survey sessions were of two types: 8 rank 2 trials as detailed in Section 3.1.1, as well as *2 rank 1* trials where participants were shown one query image and two reference images and were tasked with indicating which single reference image was more similar to the query. Example triplets extracted from this dataset are shown in Figure 3.4 (a), (b) and (c).

The reason for choosing the dataset was the limited number of categories which posed (presumably) a more difficult challenge than the ImageNet-HSJ dataset due to the heightened visual similarity of each triplet. Not only, the fact that the dataset consists of both 8 rank 2 and 2 rank 1 similarity judgements presents an interesting basis fir discussing model evaluation findings (in terms of comparing triplet - notion discussed in Section 3.2 - performance for triplets originating from 8 rank 2 or 2 rank 1 judgements).

3.1.3 BAPPS 2AFC

Berkeley-Adobe Perceptual Patch Similarity (BAPPS) dataset 2AFC [53] is very different from the datasets we introduced above. In this dataset, reference images are distortions (using various algorithms) of the query image. The list of different distortions and their description is found in [53] - Table 2.

The study is conducted on AMT as those above and the trials are of the 2 rank 1 format (where the participant is given a query image and two references which are distortions of the query image). This study involved sentinels in the survey trials (very obvious right or wrong answers - one reference with an insignificant distortion and one reference which is heavily distorted) to ensure quality of participant answers. An example of

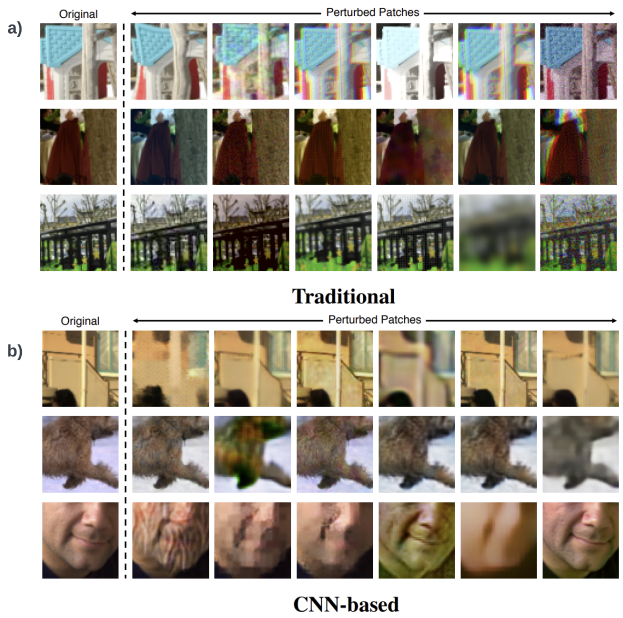

Figure 3.3: Image and study credit: [53]. (a) Traditional distortions and original image. (b) CNN-based distortions and original image.

triplets from this dataset can be seen in Figure 3.4 (f)-(k). Throughout this project, we will refer to colour, deblur, interpolation and super resolution-based distortions as real algorithm distortions alike the original BAPPS dataset study [53].

The reason for choosing this dataset was how distinct it was from the previously chosen datasets: each reference simply being a distortion of the reference - pre-existing contextual knowledge of individuals (as discussed in Section 2.1.1) is logically not likely to play a part in estimating which reference is more similar to the query; but rather, it involves ”pure” similarity judgements. Not only, the query images themselves are also very different from those of ImageNet-HSJ and Birds-16: rather than seeing a whole scene or object (a broad picture), BAPPS 2AFC only contains patches of an image (segments of a whole image).

3.2 Pre-processing

Pre-processing steps had to be taken in order to make use of the previous datasets. Firstly, triplets of images of the form (query, reference 1, reference 2) were extracted from each dataset. These triplets consist of three types, depending on the second reference image and on the collection format (i.e. 8 rank 2 or 2 rank 1), used:

- **8rank2 similar** triplet: denotes a triplet from a trial collected using an 8 rank 2 format where reference 1 is the most similar to the query, whilst reference 2 is the second most similar;
- **8rank2 dissimilar** triplet: denotes a triplet from a trial collected using an 8 rank 2 format where reference 1 is the most similar to the query, whilst reference 2 was

deemed dissimilar to the query indirectly by not being chosen from the 8 possible options as either the most or second most similar to the query. This indirectly dissimilar image has been randomly chosen for each triplet from the 6 images which have not been deemed the most or second most similar to the query;

- **2rank1 dissimilar** triplet: denotes a triplet from a trial collected using an 2 rank 1 format where reference 1 is the most similar to the query, whilst reference 2 was deemed dissimilar to the query directly by not being chosen from the only 2 possible options as similar to the query.

Each triplet set contains 1000 triplets which have been extracted from each dataset by sampling random trials. In the case of ImageNet-HSJ, only trials which have passed all four catch-trials have been chosen. Each image was resized into a 224 by 224 pixel image right before being used as input for any machine learning model (following common resizing patterns in PyTorch implementations): both the ImageNet-HSJ and the Birds-16 dataset contained images of various sizes, whilst all images of the BAPPS 2AFC dataset were 256 by 256 pixels. Figure 3.4 shows examples all resulting 11 triplet sets. The BAPPS 2AFC dataset generated 6 (the number of different distortions) 2rank1 triplet sets; ImageNet-HSJ generated 2 (similar and dissimilar) 8rank2 triplet sets; and lastly, Birds-16 generated 3 such sets: 2 of them are similar and dissimilar triplet sets from 8rank2 trials and one dissimilar triplet set generated by 2rank1 trials.

Figure 3.4: Triplet examples illustrating all triplet sets extracted from datasets (denoted on the right-hand side). Figure best viewed on screen.

Chapter 4

Methodology

The aim of this chapter is to provide a more detailed overview of the methods involved in this project. Firstly, we will discussion in-depth the neural network models and their various architectures used in this project, how these models are trained, models which train in an unsupervised manner (lacking labelled datapoints), as well as how we make use of these models. The project evaluates the ImageNet pre-trained version of the models, as well as fine tunes them on the data at hand using a special loss function - a dedicated section shall explain how this is achieved. Lastly, the concrete model architectures that have been selected shall be described and a justification provided.

4.1 Neural Networks

Whilst Section 2.3 explains neural networks on a very high level, this section aims to delve into different architectures, how one trains a neural network (and types of training), as well as how these architectures will be used and implemented as part of our experiments. The next sub-sections shall explain the main architectures employed by this project.

4.1.1 Feedfoward Neural Networks

Feedfoward Neural Networks ([16] - Chapter 6), also known as Fully-connected Neural Networks or FNN, represent the simplest neural network architecture where each node in a layer is connected to all nodes in adjacent layers (Figure 4.1 (b)). Between pairs of fully-connected layers, we see a non-linear activation functions (Figure 4.1 (b) shows them bundled together with the layers whose outputs they apply activations to). Each node to node connections is weighted - the weight parameter $w_{l,u}$ (l being the layer and u the unit number of the originating end of the connection) indicates the amount of influence that the output of a node will have on the input to the next node. A bias is also added to the product of inputs and weights before being passed to the next node. It is to be noted that weights and biases are learn-able: that means, they get adjusted gradually as our model trains (learns from examples) - this shall be discussed in Section 4.1.5.

The parameters (weights and biases) of each layer are represented as matrices: layer N

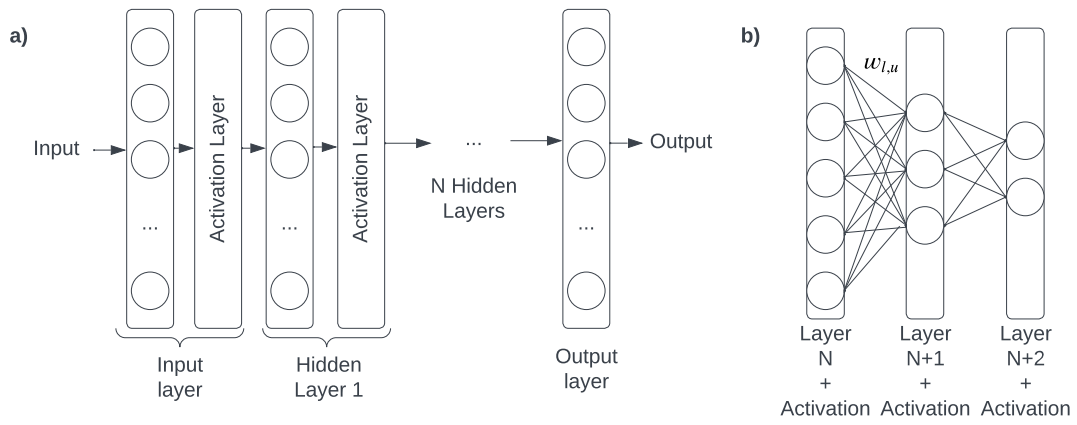

Figure 4.1: (a) A representation of a simple neural network architecture with activation layers (note: we often see activation layers tied to previous whose outputs they apply activations to). (b) A representation of a number of fully-connected layers - where each unit in a layer is connected to each unit in the following layer and each connection presents weight $w_{layer,unit}$.

weight matrix $\mathbf{W}_N$ of size $I \times O$ where I is the size (number of nodes) of layer $N-1$ and O of $N+1$ respectively; layer N bias matrix $\mathbf{B}_N$ of size $O \times 1$ where O is the size of layer $N+1$. We represent the activation function applied to the outputs of a layer N as ϕ_N and assume it to be element-wise. Data flows from one layer N to the next layer $N+1$ in a feed-forward manner (thus the name). Input $\mathbf{x}$ is propagated through a layer N using the matrix operations represented in Equation 4.1 to obtain layer output $\mathbf{z}_N$:

$$\mathbf{z}_N = \phi_N(\mathbf{W}_N^T \mathbf{x} + \mathbf{B}_N) \tag{4.1}$$

4.1.2 Convolutional Neural Networks

As introduced in Section 2.3, Convolutional Neural Networks or CNNs have been designed with the intention of processing data with a grid-like topology and have been inspired by the biological visual system [33], [23]. Their architecture is more complex than that of an FNN. CNNs present convolutional layers, pooling layers, as well as fully-connected layers as those explained in the previous section and in Figure 4.1 (b).

They have been introduced in the early 1990's and have recently been popularised in the early 2010's with the creation of models such as AlexNet [28], which won a number of competitions for its improvement compared to competitors [31] in vision tasks, including the 2012 ImageNet challenge. They became the de facto standard for various computer vision tasks.

The intuition that CNNs were built on concerns the use of hierarchical *features* and spatial relations within an image (for example, pixels close in distance may belong to the same object in an image): low-level features (edges, for example) can be combined to form higher level features (shapes), these features can be further combined to obtain an even higher level of features (objects). This intuition was inspired by a study [23] which found that neurons within the visual cortex of some animals respond selectively to certain visual patterns (features) - it was then inferred that the visual system may

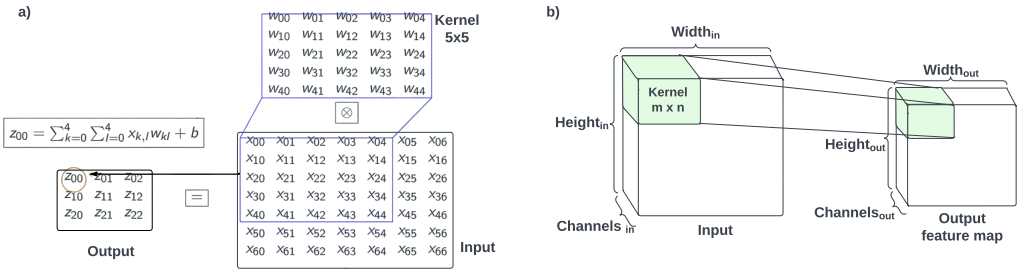

Figure 4.2: (a) A visualisation from [9] of the convolutional operation applied by a single kernel of a convolutional layer to a single channel. The $\bigotimes$ symbol represents the cross-correlation operation between two matrices. (b) A visualisation of a kernel applied to an input of dimension $Height_{in} \times Width_{in} \times Channels_{in}$.

process images in a hierarchical manner. Appendix B.4 visualises this hierarchical feature extraction in CNN architectures[1]: we see that lower level features, extracted by earlier layers, are closer to raw pixel data, whilst higher level features extracted by later layers and thus deeper into the network, have clear semantic meaning and are human interpretable.

To now understand the layers making up a CNN: a convolutional layer applies a set of filters or kernels (a matrix of a specific size aimed to capture a pattern) to the input (an image or a feature set from another layer) in order to extract meaningful features (the output of such a layer is a feature map). Each filter is then slid over the input and the dot product between the input location (the *local receptive field*) where the filter is currently applied is calculated and inserted as a single value into the output feature map. The convolution operation performed by such a layer for one single kernel is illustrated in Figure 4.2 (a). The weights a convolutional layer are the set of kernels (small matrices) applied to the input. It is to note that the matrix convolution operation is actually implemented as cross-correlation $\bigotimes$ deep learning libraries. Activation functions may reside between convolutional layers. Given a layer N and an input $\mathbf{x}$ of size $i \times j$ from a single previous unit, a layer kernel $\mathbf{w}$ of size $m \times n$, a bias term $\mathbf{b}$ and an element-wise activation function ϕ_N, the convolutional operation formula can be seen in Equation 4.2:

$$\mathbf{z}_N(i,j) = \phi_N((\mathbf{w} \bigotimes \mathbf{x})(i,j) + \mathbf{b}) = \phi_N(\Sigma_{k=0}^{m-1}\Sigma_{I=0}^{n-1} x_{i+k,j+I} w_{k,I} + \mathbf{b}) \tag{4.2}$$

Equation 4.2 shows the convolution operation for a single kernel and single channel of an input or feature map - an RGB image, for example, has three channels and a feature map may have an arbitrary number of channels (64, for example); whilst a convolutional layer has multiple kernels. This is shown in Figure 4.2 (b). This operation may thus become rather computationally expensive. We thus encounter pooling layers, which serve to downsample the spatial dimensions (width and height) of feature maps by aggregating nearby values in a feature map. Figure 4.3 provides an illustration of this operation.

[1]Figure credit: http://introtodeeplearning.com/

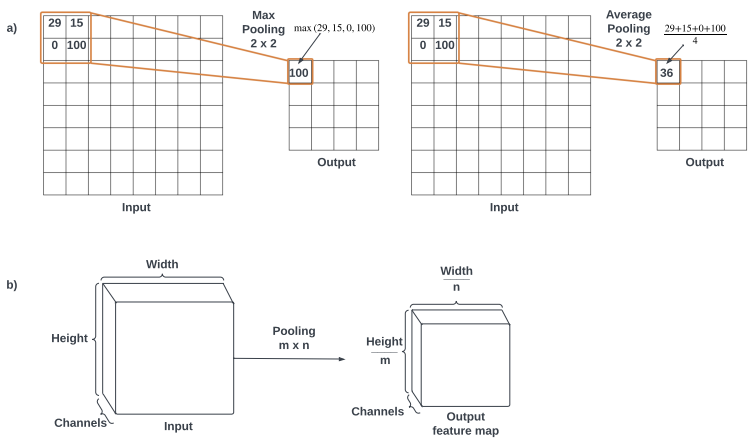

Figure 4.3: (a) A visualisation inspired by [52] of the pooling operation applied to a single channel both as a max-based and as an average-based function respectively. (b) A visualisation of how pooling affects feature map size: we notice a reduction by a factor of m and n in the height and width dimensions respectively.

Aggregation methods illustrated are based on a maximum operation (selecting the maximum from a region of arbitrary size) or an average operation (averaging values in a region of arbitrary size).

We now know the layers involved in the operation of such a network, the way that these layers are organised is by first employing a series of convolutional layers to extract the features of an image, followed by a series of fully-connected layers which make use of these features to predict an outcome (such as an image category). The full architecture is discussed in Chapter 4.

4.1.3 Residual Connections

A number of neural networks employ the use of residual connections [20], these connections allow information to flow from one layer to another without passing through intermediate layers (thus ”skipping” layers). Their purpose is to mitigate the vanishing gradient problem [15], [21] - an issue in deep neural networks where the gradual learning of the parameters halts due to the updates to these parameters becoming extremely small and thus insignificant. This problem has been a great impediment in training and thus using deeper network architectures which were able to solve complex tasks. ResNet [20] (discussed in more detail in Chapter 4) is a CNN architecture (winner of the 2015 ImageNet competition) which employs residual connections.

4.1.4 Transformers

As mentioned in Section 2.3, Transformers [50] are neural networks which employ attention mechanisms to weigh the importance of different input components and have been designed with sequences in mind (such as variable sequences of words), not for single datapoints (such as a single image).

They have been created with Natural Language Processing, and successfully became the standard for such tasks. However, an approach called the Vision Transformer (ViT) [8] has introduced the use of transformers in image-processing tasks and not only, ViT was shown to attain comparable results to those of well-performing CNN-based models. Our project makes use of ViT only (from the Transformer architecture), when we refer to transformers from this point forward, we are directly referring to ViT. One may wonder how an architecture designed specifically for sequential data can be adapted to single grid-like datapoint - images: this is done by first dividing the image into fixed-size patches, which are then processed by the network one by one.

The components that make up ViT are: patch embedding, feature extraction and classification head. We shall start with patch embedding (represented in Equation 4.3 - [8] - Equation 1): the model takes an input image $\mathbf{x}$ of size $H \times W \times C$ as input (C is the number of channels), this image is then split into N square patches of shape $p \times p \times c$ (p is pre-defined) from top to bottom, left to right. These patches are then flattened to N line vectors of shape $1 \times (p^2 * c)$. These line vectors are then multiplied by a learn-able (not preset - training is introduced in Section 4.1.5) embedding vector $\mathbf{E}$ of shape $(p^2 * c) \times d$ (d is preset) which outputs N embedded linear patches of dimension $1 \times d$. A token $\mathbf{x}_{class}$ of dimension $1 \times d$ which represents an aggregate of all patch representations [7] is then added at the beginning of the N embedded linear patches. A learn-able matrix $\mathbf{E}_{pos}$ of size $(N+1) \times d$ (which learns positional information for each patch) is added (element-wise) to the concatenated N embedded linear patches which results in the output of the patch embedding component z_0.

$$\mathbf{z}_0 = \left[\mathbf{x}_{class}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; ...; \mathbf{x}_p^N\mathbf{E}\right] + \mathbf{E}_{pos},\ \mathbf{E} \in R^{(p^2 \cdot C) \times d}, \mathbf{E}_{pos} \in R^{(N+1) \times d} \tag{4.3}$$

The feature extraction component then takes the concatenated patch embeddings $\mathbf{z}_0$ and learns features from them (Equation 4.4 - [8] Equation 2,3). This component is made up of a stack of L transformer enconders. The enconder components contain a multi-headed self-attention (*MSA*) mechanism, a 2-layer FNN (*MLP*), residual connections [20] (Section 4.1.3); as well as normalization layers *LN* (these simply scale with the mean and standard deviation for each training example to reduce training time [2]). The multiheaded sel-attention mechanism ([8] - Appendix A) divides an input sequence into smaller sequences called ”heads” and calculates self-attention (essentially a weighted sum of the values associated with all the positions in the input sequence, the weights being computed based on the similarity between a given position and all other positions in the sequence) for each head, which captures relationships between elements in the sequence in parallel.

$$\mathbf{z}_l = MLP(LN(\mathbf{z}'_l)) + \mathbf{z}'_l \text{ where } \mathbf{z}'_l = MSA(LN(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1} \text{ and } l = 1...L \tag{4.4}$$

Lastly, the classification head component is applied only to the last representation (output of the feature extraction component) of the $\mathbf{x}_{class}$ token (Equation 4.5 - [8] Equation 4). The classification head component is a simple FNN. Figure 4.4 illustrates how all of these components come together to build the ViT architecture.

$$\mathbf{y} = LN(\mathbf{z}_L^0) \tag{4.5}$$

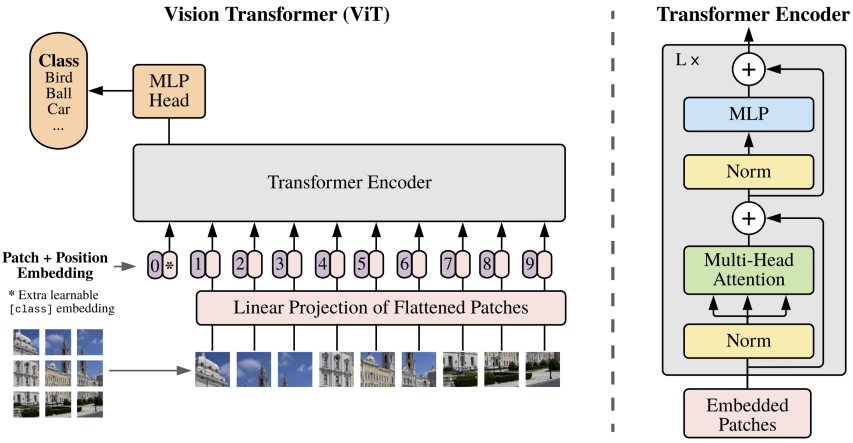

Figure 4.4: A visualisation from [8] of the Transformer architecture depicting the three aforementioned components: patch embedding serving as input to the Transformer enconder, the Transformer encoder and lastly, the (MLP - stands for Multi-layer perceptron and is equivalent to FNN) classification head.

4.1.5 Supervised Training

The network attempts to ”learn” the correct mapping between an input and output for a datapoint by adjusting network parameters, such as weights and biases, gradually - this procedure is called training. The network parameters usually start from random values. In supervised training ([16] - Chapter 5.7) or learning, we work with labelled training data - that is, we know the correct output and use it to align model output values to expected output values. The goal is to learn the input-output value mapping. Training is done by showing the network a collection of N training datapoints $\mathbf{x}_i$ for $i \in [0...N]$ for which we know the correct output $\hat{\mathbf{y}}_i$ (a vector of dimension d): for each datapoint, the network will predict an output $\mathbf{y}_i$ (it will perform a forward pass) which will serve as input to the network's loss function. The loss function calculates the quality of the network's prediction (how close it is to the expected output). There are many loss function options ([16] - Chapter 8) - however, a very simple and intuitive example is the Mean Squared Error loss function shown in Equation 4.6 - we can clearly see that MSE simply calculates the distance between the vectors representing the expected and actual output. MSE is typically used for regression tasks.

$$MSE = \frac{\Sigma_{j=1}^{d} \left(y_j - y_j\right)^2}{d} \tag{4.6}$$

If we are working with a binary classification task, with N datapoints, where we predict the probability $\hat{y}_i$ of a datapoint i belonging to one of two classes, and we know the true label for that datapoint y_i, where $y_i = 0$ or $y = 1$, we would instead use Binary Cross-Entropy Loss as shown in Equation 4.7

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \tag{4.7}$$

The output of the loss function will then be used by the network to update its weights (the amount of an update is established by a variable called the learning rate α) by

calculating the gradients of the parameters with respect to the loss function [40] using the chain rule of calculus - this process is called backpropagation. The parameters of the network will then be updated using an optimization algorithm which gradually adjusts the weights and biases in the direction that minimizes the loss. In other words, updating the parameters in the direction of the negative gradient of the loss function with respect to the parameters.

Training is an iterative process, the above procedure being repeated for each input/set of inputs in the training set multiple times until the loss is minimizes to an acceptable level. An epoch is a complete iteration over the entire dataset.

The two optimizers used by the models evaluated in this project are Stochastic Gradient Descent [25, 16] (SGD) and Adaptive Moment Estimation [26] (Adam).

SGD uses a randomly (thus Stochastic) selected subset of the training data to compute the gradient at each iteration - this randomness allows SGD to escape from local optima and possibly converge to a better solution. Given network parameters at iteration t, θ^t, learning rate α and $\nabla_\theta Loss(\theta; x_i, y_i)$ as gradient of the loss function with respect to the model parameters evaluated at the i-th datapoint; SGD formula at single example can be seen in Equation 4.8.

$$\theta_t = \theta_t - \alpha \nabla_\theta Loss(\theta_t; x_i, y_i) \tag{4.8}$$

Usually, momentum is also often added to the use of SGD to accelerate and smooth out convergence towards a minimum by adding some fraction of the previous update vector to the current update vector. We use Nesterov momentum [45] due to its improved performance when compared to classical momentum for DNN - the modified formula for SGD with Nesterov momentum is presented in Equation 4.9. In the aforementioned equation, v_t is the Nesterov momentum vector at iteration t, γ is the momentum coefficient and $\theta_t - \gamma v_{t-1}$ is the look-ahead point. Nesterov momentum takes a step in the direction of the momentum vector before evaluating the gradient - it thus evaluates the gradient at a point that is ahead of the current position in the parameter space (the look-ahead point).

$$\theta_{t+1} = \theta_t - v_t \text{ where } v_t = \gamma v_{t-1} + \alpha \nabla_\theta Loss(\theta_t - \gamma v_{t-1}) \tag{4.9}$$

Adam is an extension of SGD and it computes individual adaptive learning rates for each parameter based on estimates of the mean of the gradients (called the first moment of the gradient m_t) and the variance of the gradients (called the second moment of the gradient v_t). These estimates are moving averages of the gradients and square of the gradients and they are computed using two preset decay rate parameters (one for each moment β_1 and β_2 respectively): which specifies the amount of weight of past gradients to the current estimate.

The model parameters at some iteration t, θ_t are updated using the adaptive learning rates and the gradient at t, g_t. To note, $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected estimates of the first and second moments and ε is a small constant used to mitigate division by zero. Adam updates parameters following the formula in Equation 4.10 - full formula

in Appendix B.2.

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{4.10}$$

Due to this mechanism for adapting the learning rate to the geometry of the loss surface, Adam presents improved speed and stability of convergence over SGD, however, also due to this adaptive characteristic, Adam is more prone to overfitting ([16], Chapter 5, Figure 5.2) to small datasets - that is, learning the intricacies and noise of the training dataset which do not in fact exist in another dataset, thus, when the model is tested on unseen data, it tends to yield suboptimal performance. The choice of an optimizer depends on the model and the dataset of choice (no ”best” optimizer).

4.1.6 Unsupervised and Self-supervised Training

Opposite to supervised learning, **unsupervised learning** ([16] - Chapter 5.8) uses unlabelled training datapoints - that is, we do not know the correct output that an input maps to. The goal of unsupervised learning is to identify and learn patterns and underlying structural information within the set of datapoints. This can be achieved by grouping similar datapoints or identifying outliers in the training set, for example.

We use the **K-Means** clustering [19] unsupervised machine learning algorithm (not a neural network) in this project. The algorithm partitions a dataset into K clusters based on their similarity - a popular similarity measure is Euclidean distance. K is a constant picked by us, C is the set of K clusters and x a datapoint. The algorithm considers each image as a matrix of values (a 2D array in Python). It first selects K datapoints from the dataset as the initial centroids, μ, of a cluster and then assigns each datapoint in the training set to the closest centroid based on a distance metric such as Euclidean distance. We now obtained our initial K clusters. We then obtain the mean of each cluster and update the centroids to be these means. The procedure described is then repeated a number of times until convergence - namely, until the centroids' values stop changing. This procedure is illustrated in Equation 4.11. The algorithm strives to minimize the sum of the squared distances between datapoints in a specific cluster and the cluster's centroid. Appendix B.1 illustrates this procedure.

$$KMeans(C,\mu) = \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2 \text{ where } C = [C1, C2, ..., Ck] \tag{4.11}$$

Self-supervised learning is a type of unsupervised learning where the model uses the input datapoints to generate its own outputs/labels. In the absence of labelled data, the network essentially creates its own dummy tasks, also called pretext or proxy tasks, that map inputs to outputs, as generated from inputs. These tasks, although not the task we will apply the network to, will force the network to learn meaningful representations of the input data, thus allowing us to use feature extraction layers for our task. This pretext task can be achieved in many ways such as rotating the input images and asking the network to predict the angle of rotation, or by obscuring some patch of the image and asking the network to predict the missing patch. The chosen approaches for this project

includes **Jigsaw Puzzle** [35] task due to its performance, situated on the high end when compared to other self-supervised learning paradigms, when applied to the BAPPS dataset - as presented in the original study [53]. The Jigsaw puzzle solving approach is done by splitting the original input image into patches which are then shuffled. The network is then asked to predict the correct ordering of these patches, allowing the network to understand the local features of an image.

4.1.7 Validation

We now know that a portion of the datapoints that we have are set aside and used for training the model. However, we must make sure that we do not train the model to the point of overfitting to the training data. To mitigate overfitting, we set another portion of our dataset aside, typically much smaller than the training portion, called the validation set. Using the validation set, we test our model's predictions after each training epoch to prevent overfitting. A clear sign of overfitting is the validation loss increasing whilst the training loss still decreases. This indicates that we should stop training at that point or modify one of the hyperparameters of the model - such as the learning rate or the number of training epochs.

4.2 Evaluation and Fine-tuning Frameworks

We have now presented all neural networks and machine learning algorithms used in this project. This section aims to explain how the models will be used in our experiments, namely: how we will evaluate pre-trained neural networks and algorithms on the sets of image triplets described in Chapter 3, and how the pre-trained neural networks will be fine-tuned - thus trained on the image triplets.

4.2.1 Feature Extraction and Classification

We can think of most neural networks as split into two components: a feature extraction [36] and a classification component. The feature extraction component refers to the earlier layers of the model which are responsible for learning meaningful representations of the input data, namely features. These features are then given as input to the classification component, made up of the final layers of the network which are typically fully-connected layers with activation functions, which will make a prediction, the final model output, based on these features. The prediction can be the class that an image belongs to, or a vector meant to represent the image in an N-dimensional space, for example. An illustration of this split is shown in Figure 4.5 (a).

4.2.2 Evaluating Pre-trained Models

We will make use of the concept of transfer learning [5] in this project: that is reusing a pre-trained neural network on a new task to take advantage of the fact that DNNs learn hierarchical representations of data, and thus the feature extraction component can be repurposed. PyTorch offers pre-trained models on the ImageNet dataset, trained on the task of supervised image classification on the 1000 categories of ImageNet.

This is highly beneficial as we can re-use the parameters learnt from extensive training on a very large dataset, without having to train the models ourselves due to limited computing resources.

The following pre-trained model architectures have been chosen[2]:

- **ResNet** (Residual Network) [20] is a family of CNN models that make use of residual connections to mitigate the vanishing gradient problem that DNNs are prone to. Their architecture is in fact very similar, layer-wise, to that of the dummy architecture presented in Figure 4.5 (a). The reason for choosing this architecture is its proven high accuracy at predicting human estimates of image similarity when tested on the ImageNet-HSJ dataset: the study [38] shows ResNet architectures achieving the highest performance.
- **EfficientNet** [47] is a family of CNN models that have been designed to achieve state-of-the-art image classification accuracy whilst being computationally efficient - for example, they manage to achieve comparable performance, and even surpassing, accuracy to very deep variants of ResNet. EfficientNet essentially introduces a scaling parameter which uniformly scales the network width, depth and resolution depending on the level of complexity required. These networks make use of the same layers as ResNet models, with the addition of using depth-wise and point-wise convolutions, first introduced in MobileNet [22], which represent a more efficient way, computationally and memory-wise, of applying convolution layers at a small accuracy cost. The reason for choosing to explore EfficientNets is due to their complexity scaling procedure - it may be interesting to observe is this scaling procedure will cause model complexity and accuracy (on our task) to be correlated as shown in Figure 1 of the original study [47]. It is also due to the fact that it is not truly explored in other studies relavant to the project [38, 30, 53].
- **MobileNet-V2** [42] is a CNN model designed for mobile and embedded devices with limited computational power. It uses all layers used by EfficientNet, with the addition of inverted residuals, which are a more efficient way of applying residual blocks (a set of layer delimited by two ends of a residual connection) - at a small reduction in accuracy. This model is of interest to our project due to its computational efficiency, being the smallest model, in terms of the number of parameters, explored in this project.
- **AlexNet** [28] was a ground-breaking CNN model at the time of its appearance, winning ILSVRC in 2012 due to its performance on ImageNet. It only presents 8 layers, being the shallowest model explored in this project. The architecture makes use of traditional convolutional and fully-connected layers. Despite its simplicity and the fact that it is the least recent neural network from the ones we list enumerate, its performance on BAPPS finds itself at the higher end - being the reason for wanting to use the architecture in this project.
- **ViT** (Vision Transformer) [8] is a Transformer architecture which has been discussed previously in this chapter. The reason for wishing to explore this

[2]A list of all available PyTorch pretrained models can be found at: https://pytorch.org/vision/stable/models.html

architecture is due to the multiple studies [48, 11, 13] claiming that ViT is closer to human perception than CNN architectures, despite the fact that CNNs were inspired by the biological visual system as discussed in Section 4.1.2.

To observe the correlation between model complexity, number of layers/parameters, and accuracy at predicting human estimates of image similarity, multiple complexity variants of each of the architectures described above will be evaluated in this project - with the exception of AlexNet and MobileNet due to a lack of alternative architectures. By complexity variants, we refer to the fact that some of the architectures above may contain a different number of layers: from shallow networks to very deep networks. Namely, for ResNet we will evaluate ResNet 18, 34, 50, 101 and 152; for ViT that is ViT B 16 and 32, as well as ViT L 16 and 32 (”B” and ”L” stand for base and large respectively); lastly, for EfficientNet that is B0 through to B6.

The manner in which these pre-trained models will be evaluated is using the feature extraction/classification component separation depicted in Figure 4.5 (a): we will input an image and obtain as output the feature representation - originally a set of matrices, however we flatten it into a feature vector - outputed by the last layer of the feature extraction component. We will perform this procedure for each image in a triplet in parallel - Chapter 3 describes the triplet formation. We will then use cosine similarity to compare the feature vector of each reference compared to the query: the reference image feature vector for which the cosine similarity, when compared to the query feature vector, is the highest is considered the decision of the network in terms of choosing a reference most similar to the query image. This procedure is shown in Figure 4.5 (b). The reason for using cosine similarity instead of Euclidean distance as a vector similarity measure is due to the fact that it is not affected by vector magnitude - it is also used by other studies relevant to this project [53, 38].

In terms of how the K-Means algorithm will be applied to our image triplets: we will fit K-Means, K=2, clusters to the reference images using cosine similarity as a distance metric. We will then give the algorithm the query image and see which reference's cluster it is assigned to - this shall represent the decision of the algorithm for that triplet.

4.2.3 Fine-Tuning Pre-trained Models

Alike the original BAPPS dataset study [53], we will also tune our pre-trained models: that is, we will train these models, starting from their pre-trained parameters, on the triplet datasets that they will be applied to. We will choose the top four performing architectures from above in terms of their average accuracy on all triplet sets. From these architectures, the complexity variant (where applicable) will be chosen based on available computing resources (for example: if ResNet-50 and ResNet-101 present a difference of accuracy at below 1% then ResNet-50 shall be fine-tuned). Fine-tuning will be done in both a supervised and unsupervised manner.

As described in Chapter 3, we obtained 11 triplet sets of 1000 triplets each. We will split these sets into training, validation and testing sets based on the popular splitting pattern [16] of 80%/10%/10%: for each triplet set, the training set will have 800 triplets, validation and testing sets will have 100 each. It is the accuracy of the fine-tuned

model on the test set that will be considered when evaluating results. For triplets from the Birds-16 dataset, where we know the species labels, we first ensure a balanced representation of all 16 species in each of the train, validation and test set.

Fine-tuning details for each model using the presented methods can be seen in Table 4.1 - additional implementation details found in Appendix B.3.

4.2.3.1 Supervised Fine-tuning

Supervised model fine-tuning will make use of the loss function Triplet Loss [4], available in PyTorch as Triplet Margin Loss. After our model f outputs the feature vector representations of our three images, we will input them into the Triplet Loss function. The query image feature vector is called the anchor A, the most similar reference is called the positive P, and the second most similar/dissimilar reference is called the negative N. The Triplet Loss function aims to minimize the distance between the anchor and the positive while maximizing the distance between the anchor and the negative. A margin hyperparameter indicates the minimum required separation between the positive and negative distances. This procedure can be seen in Equation 4.12:

$$TripletLoss(A,P,N) = \max(|f(A) - f(P)|^2 - |f(A) - f(N)|^2 + margin, 0) \quad (4.12)$$

To fine-tune pre-trained models using supervised learning we will take the feature extraction component of the pre-trained model and add our own fully-connected block at the end of it (we will call this the **linear component**) - essentially creating a new model. The purpose of the linear component is to take the feature extraction component features and embed them in an N-dimensional space. N was chosen to be 32; less than the highly dimensional feature extraction component output, yet containing enough dimensions to reflect the complexity of our data - a trial and error approach was used for N values ranging from 10 to 100, 32 yielding best accuracy results.

Supervised fine-tuning will be done in two ways (similar to the approaches in the original BAPPS dataset study [53]): **(1) Frozen approach:** we keep the feature extraction component parameters from the pre-trained model unchanged, we "freeze" the parameters, and we only train the linear component; **(2) Unfrozen approach:** we train both the feature extraction and the linear component. An illustration of this framework and how supervised training will proceed is shown in Figure 4.5 (c).

We will test the resulting tuned models in two ways: by using the entire new model to generate an image vector output - we will thus test the resulting vector from the linear component; and by only testing the tuned feature extraction component output in an identical manner to the illustration in Figure 4.5 (b).

4.2.3.2 Self-supervised Fine-tuning

We use the self-supervised Jigsaw Puzzle method described in Section 4.1.6 to fine-tune our models. To do so, we will use the pre-trained model's feature extraction component and add our own component on top of it in order to match the pretext task requirements. The models will be trained on the query images of the triplets in the training set and on modification of the query images depending on the pretext task.

We will first take each query image in the training set, split it into 9 patches - a 3 by 3 grid - inspired original study [35] - and scramble those patches to form a new image which will serve as input to our network, whilst the original query image becomes the expected output. We now have our training and validation input-output pairs for the pretext task.

We shall now form our model by using the pre-trained model feature extraction component and a new final linear component with output size equal to the number of patches - 9. The output of the model, of dimension 9, predicts the correct order of the scrambled patches to form the expected output. This can be seen as a classification task, where each patch is a class label. A popular choice for classification tasks loss functions is Cross Entropy loss - which is our choice for this model. The Binary Cross Entropy loss from Equation 4.7 can be adapted to be used on multi-class classification.

To test the models after self-supervised fine-tuning, we test the tuned feature extraction component output in an identical manner to the illustration in Figure 4.5 (b). We do not test the entire network including the linear component for the pretext task due to the lack of a link between the pretext and actual task which will not make for a fair comparison with supervised fine-tuning of models.

Model	**Learning**	**Freezing FE**	**Learning Rate**	**Epochs**	**Batch**
AlexNet	Supervised	Frozen	0.01	15	32
AlexNet	Supervised	Unfrozen	0.01	15	32
AlexNet	Self-supervised	N/A	0.01	15	32
ResNet-50	Supervised	Frozen	0.01	15	32
ResNet-50	Supervised	Unfrozen	0.001	20	16
ResNet-50	Self-supervised	N/A	0.001	20	16
ViT-B-32	Supervised	Frozen	0.01	15	32
ViT-B-32	Supervised	Unfrozen	0.0005	20	32
ViT-B-32	Self-supervised	N/A	0.0005	20	32
EfficientNet-B3	Supervised	Frozen	0.01	15	32
EfficientNet-B3	Supervised	Unfrozen	0.01	20	64
EfficientNet-B3	Self-supervised	N/A	0.01	20	64

Table 4.1: Implementation hyperparameter details of each fine-tuned model. Hyperparameters have been chosen on trial and error basis, as well as based on the hyperparameters used by the PyTorch team when training the pre-trained models on ImageNet. FE refers to Feature Extractor.

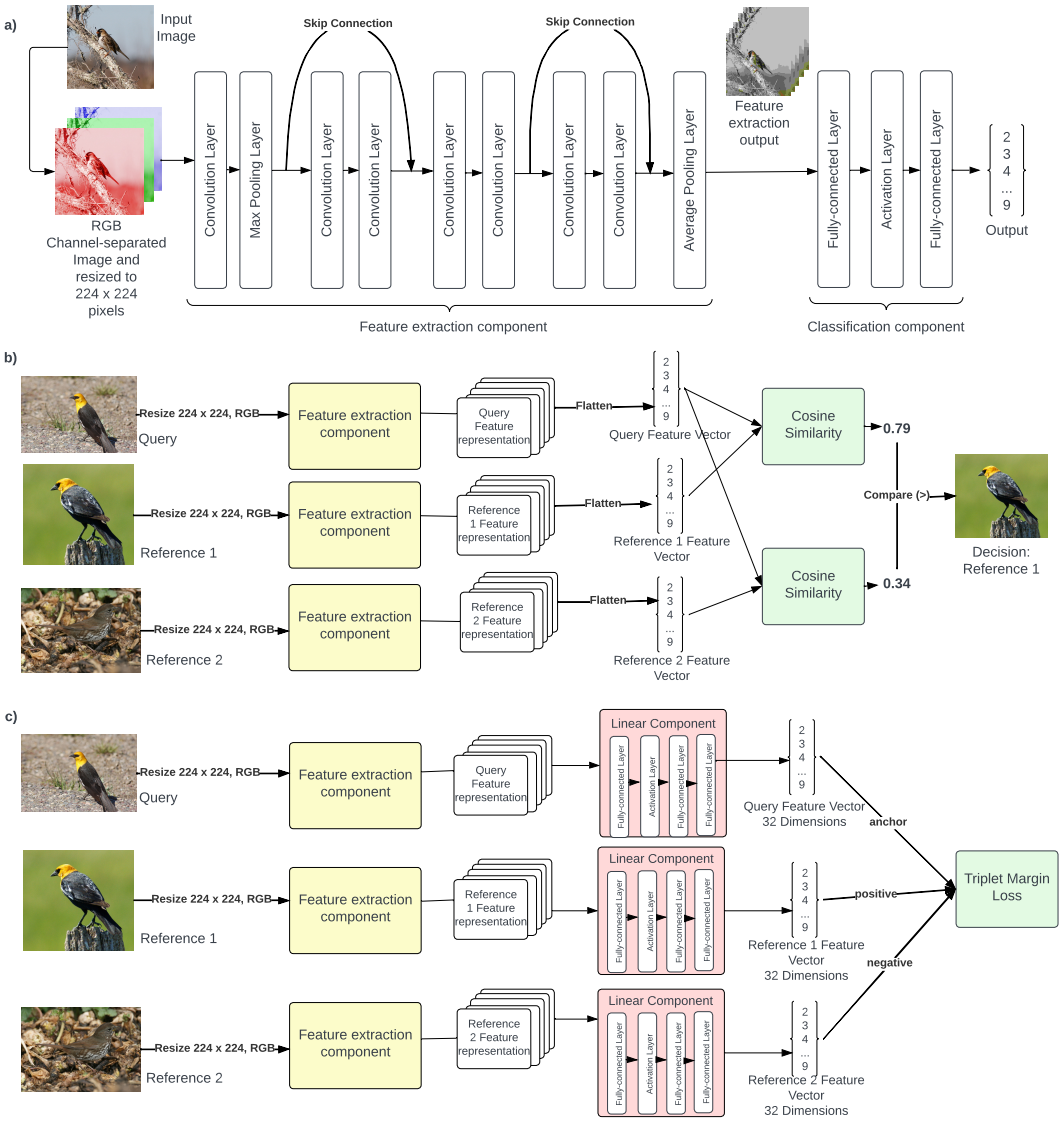

Figure 4.5: An illustration of how pre-trained models have been evaluated and fine-tuned. (a) Shows the architecture of a DNN - similar to a CNN (although the same feature extraction/classification component split exists for ViT) and how it can be split in two components: feature extraction and classification. (b) Shows how a pre-trained model will be evaluated on the triplet prediction task. (c) Shows the framework used in fine-tuning a pre-trained model using supervised learning. Best viewed on screen.

Chapter 5

Experiments and Results

The pre-trained models and the K-Means algorithm presented in Section 4.2.2, and the fine-tuned models described in Section 4.2.3.1 have been implemented and evaluated for the purpose of conducting the experiments described in this chapter. This chapter aims to describe the purpose of the experiments, their hypotheses, the way in which they were conducted, and an interpretation of findings in light of the existing literature. Experiment results will either indicate approval or disapproval of the hypotheses. Experiments have been inspired by past studies [30, 53]; however, they aim not to copy already conducted experiments, but rather explore a different angle or extend upon existing experiments.

5.1 Experiments

This section shall detail each experiment and its results. The task on which models have been evaluated is triplet similarity prediction as described in Section 4.2. Final results are shown in Figure 5.4. We will refer to triplet sets by the names defined in Figure 3.4. It is to be noted that certain triplet sets have been grouped together when summarizing findings due to identical results and low informational value: Birds-16 2rank1 and 8rank2 have been grouped; traditional and CNN-based distortions, and real-algorithm-generated distortions have been grouped as well (similarly done as in original BAPPS study [53]). Following initial evaluation of pre-trained models and K-Means, the 4 top-performing pre-trained models have been chosen and fine-tuned to obtain further results - also present in Figure 5.4. All datasets and triplet sets mentioned in the discussion have been introduced in Chapter 3.

5.1.1 Complexity-Accuracy Correlation

***Hypothesis**: Larger capacity models better align better with human similarity judgements.*

This experiment aims to observe the existence of positive correlation between the capacity of an ImageNet pre-trained model, and the accuracy that the same model achieves on our triplet sets. This is inspired by a recent study [30] which evaluated the correlation between model capacity and perceptual score on the BAPPS dataset.

The study [30] found negative correlation between pre-trained model capacity and its accuracy on the BAPPS dataset, however, we note that the BAPPS dataset is extremely different from the ImageNet dataset that models have been pre-trained on.

We thus extend upon this experiment by considering different datasets with their own similarities and differences to ImageNet. ImageNet-HSJ being formed of ImageNet validation set images (most similar to ImageNet); Birds-16 images may be very specific as to what they contain, however, they are quite similar to ImageNet images - a singular object in a scene; however, BAPPS 2AFC is quite different as it contains patches of images and distortions.

To do this, we plot a complexity measure, GFLOPS (measure of floating-point arithmetic operations) or the number of parameters, against the accuracy achieved by each pre-trained model for each triplet set. We also calculate the correlation coefficient between the two variables. We can see the results presented in Figure 5.1. Either the number of GFLOPS or parameters chosen as the explanatory variable will show the same correlation trend - the equivalent figure using parameter number as a complexity measure is present in Appendix C.1. We observe very weak positive correlation and very weak negative correlation for the BAPPS triplet sets - we may interpret this as no true correlation between ImageNet pre-trained model capacity and performance on BAPPS dataset-generated triplets. As a note, the weak positive correlation value for the traditional and CNN-based triplet sets sometimes appears due to our random sampling of 1000 triplets from the BAPPS dataset - over a number different random samplings, the correlation oscillated between very weak positive and negative values. Interestingly however, we observe positive correlation between ImageNet pre-trained model capacity and the accuracy achieved on the ImageNet-HSJ and Birds triplet sets. Not only, it appears that this correlation is further heightened by triplet set similarity to ImageNet - HSJ presenting the highest correlation.

We may thus say that the experiment both disproves, if the dataset used in pre-training models is extremely different to the one that models are applied to in the context of this task, and approves of the hypotheses if the opposite holds.

This experiment may also be approached from the angle of evaluating fine-tuned models and observing whether higher-capacity fine-tuned models obtain better performance. We would expect so due to increased explanatory power and the results confirm this assumption - and thus approve the hypothesis. However, this approach is of little value in this context as we would like to explore emergent perceptual properties of models.

5.1.2 Architecture-Accuracy Correlation

***Hypothesis**: Specific architectures better align with human similarity judgements.*

The hypothesis is based on the findings of past studies [1, 48] which indicate better alignment between Transformers and human visual perception when compared to CNN models. This experiment aims to expand upon this observation by comparing CNN and Transformer performance at predicting human estimates of image similarity both when fine-tuned using various methods, as well as when pre-trained on a different task.

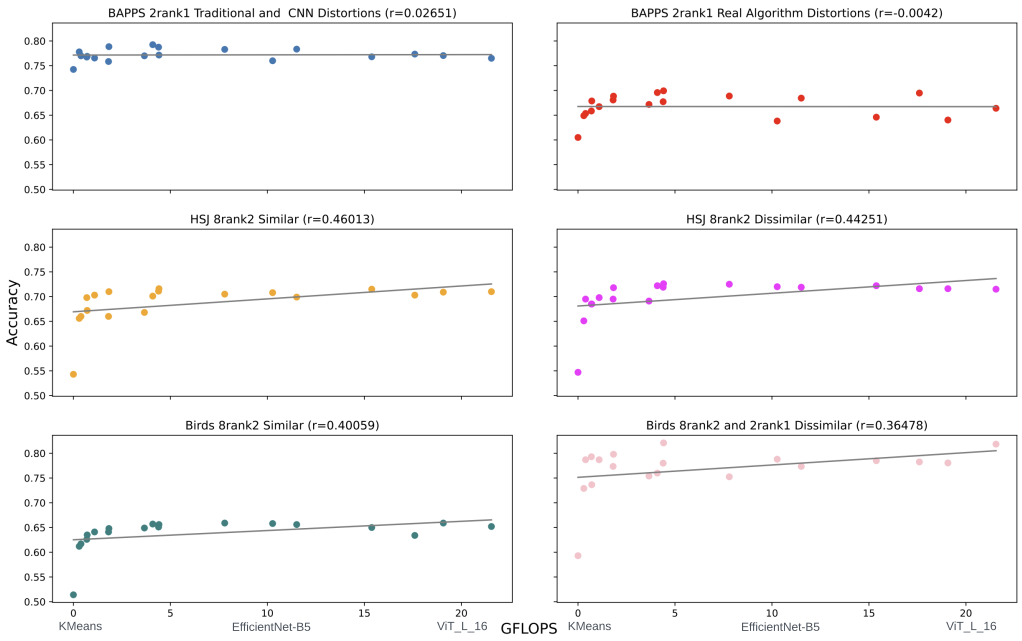

Figure 5.1: Scatter plot and correlation between number of GFLOPS of a pre-trained model and its accuracy on triplet set similarity prediction.

The results present in Figure 5.4 approve the hypothesis. When looking at pre-trained model performance, we observe that ViT models tend to attain accuracy values situated at the high end - however, this may partly be due to the heightened capacity of the ViT-L transformers (highest number of GFLOPS by a very high margin). Instead, we shall observe the fine-tuned versions of the 4 top-performing models overall. In this case, ViT-B-32 presents a comparable number of parameters to AlexNet and a comparable number of GFLOPS to ResNet-50, which allows for a more fair comparison. We observe how no matter the fine-tuning method, ViT-B-32 outperforms the other CNN models by a significant amount.

5.1.3 Transfer Learning Accuracy Correlation

***Hypothesis**: Models achieving high accuracy on pre-training classification task will also achieve high accuracy on triplet human similarity judgement prediction task.*

This experiment aims to test the existence of positive correlation between pre-trained model accuracy on ImageNet classification and the accuracy obtained by the pre-trained model at predicting human estimates of image similarity based on our triplet sets. Results are presented in Figure 5.2. We see that the majority of triplet sets present very weak correlation (positive or negative) between the accuracy achieved by a pre-trained model on ImageNet and how that translates to performance on predicting triplet similarity. Surprisingly, ImageNet-HSJ, most similar to ImageNet - which is used to pre-train the models, presents very weak correlation when compared to other triplets.

The results may indicate a lack of correlation between pre-trained model ImageNet accuracy and the accuracy at predicting similarity triplets, thus disproving the hypothesis.

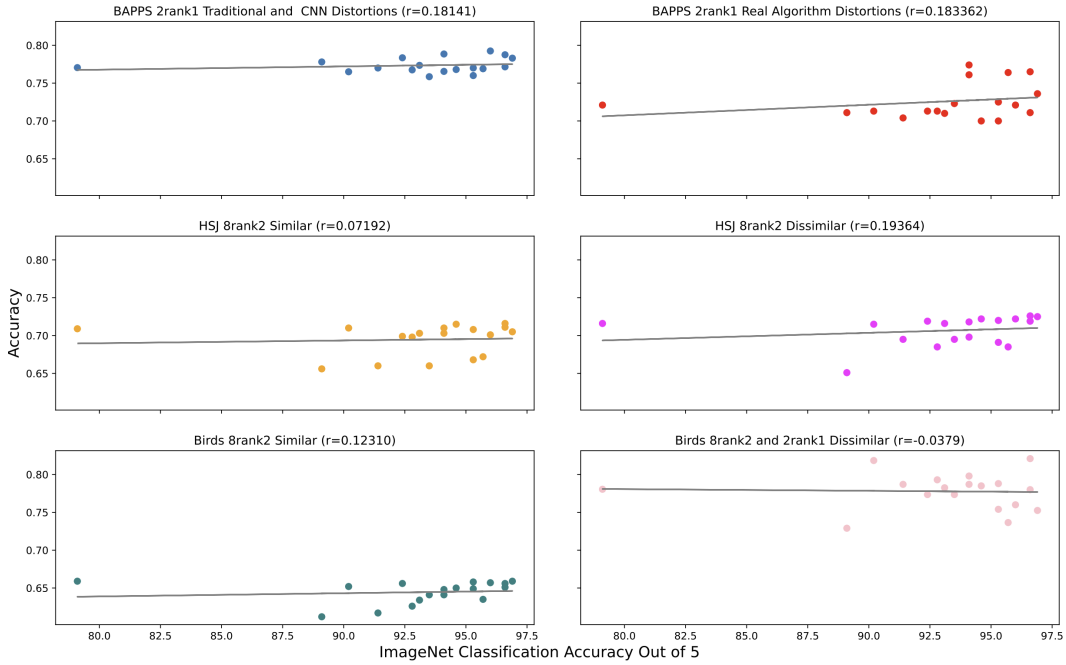

Figure 5.2: Scatter plot and correlation between pre-trained model accuracy on ImageNet classification task and accuracy on triplet set similarity prediction for each triplet.

5.1.4 Supervision Type

***Hypothesis**: Supervised training and fine-tuning of models yields better accuracy than unsupervised methods.*

This experiment aims to extend the experiments performed in the original BAPPS study [53] which only make use of supervised fine-tuning methods for ImageNet pre-trained models (very similar to our own approach). Despite having access to labelled data - in the sense, we do know the correct input to output mapping - the exploration of self-supervised fine-tuning may be of interest due to shown competitive performance on vision tasks of self-supervised models trained on ImageNet compared to their supervised adversary [18].

Results are shown in Figure 5.4: we can easily observe the large gap between pre-trained supervised models and the unsupervised algorithm, K-Means. K-Means is the lowest scoring model present in each subplot, its performance being only a few figures above random change for similar HSJ and Birds-16 triplets. Another gap is that between supervised and self-supervised fine-tuning: the gap is larger when compared to unfrozen supervised tuning. An interesting observation that we could however make is the fact that the tuning method gap tends to be smaller for similar triplets rather than dissimilar triplets - applicable for the HSJ and Birds-16 triplet sets. The experiment thus approves the hypothesis.

5.1.5 Freezing Feature Extraction

***Hypothesis**: Freezing feature extraction component whilst fine-tuning will yield worse results than unfrozen approach.*

The interest presented in experimenting with frozen and unfrozen approaches is the fact that we may be able to see what it is that makes a difference when fine-tuning pre-trained models to achieve better human judgement similarity prediction: is it the features themselves (feature extraction component) or is it how we interpret and classify these features (linear component)?

The results shown in Figure 5.4 approve of the hypothesis: in most cases, fine-tuning only the linear component does result in lower accuracies and improvements from the base pre-trained model. However, we do observe an interesting pattern: the gap between the frozen and unfrozen fine-tuning approach is smaller for lower-capacity models such as AlexNet. This may simply be due to the fact that AlexNet itself has limited learning power due to its number of parameters, whether the feature extraction component is frozen or not.

5.1.6 Collection Process

***Hypothesis**: Models will perform better on dissimilar triplets where the dissimilar reference is directly judged as dissimilar - 2 rank 1 collection - rather than indirectly by not being chosen - 8 rank 2 collection.*

This is an unexplored use of the Birds-16 dataset to the best of the writer's knowledge, despite the fact that it contains dissimilar triplets collected using both 8 rank 2 and 2 rank 1 methods. In 8 rank 2 collection, when forming a dissimilar triplet, we use an indirect judgement of dissimilarity - the dissimilar reference is deemed to be dissimilar from the query by not being chosen out of 8 options; in 2 rank 1 collection, such a triplet is formed using direct judgement - there is only 2 options.

The actual results disprove of the hypothesis: the accuracy obtained for the indirectly and directly judged dissimilar triplets is almost identical (see Appendix C.2) - it is also the reason why it has been decided to merge the two triplet sets in the final results Figure 5.4.

5.1.7 Perceptually Difficult Tasks

***Hypothesis**: Perceptually difficult tasks for humans, where both references are similar to query, translates to lower accuracy for models.*

This is again an unexplored angle to the best of the writer's knowledge: in human perception, it is logical to assume that if given three images, one query image and two reference images, and asked to estimate the most similar reference image to the query image, we may be confused if both reference images are equally similar or dissimilar to query image. Our "confusion" may thus increase with the decrease in a difference between some similarity measure taken between the query and each reference. To draw a parallel between our confusion and the way a model may be confused: the model's accuracy may decrease as the magnitude of the cosine similarity difference between the feature vectors of the query and each reference decreases (equation at the bottom of Figure 5.3.

Figure 5.3 approves the hypothesis and shows how there is positive correlation between

the difference of the predicted feature vectors of the query and each reference and the accuracy of a model on each triplet set. Interestingly, the correlation does not appear to be stronger or weaker for models with higher/lower capacity. If higher capacity models presented stronger correlation then it may have showed that higher capacity models are aligned to this human vision characteristic more so than their lower-capacity counterpart.

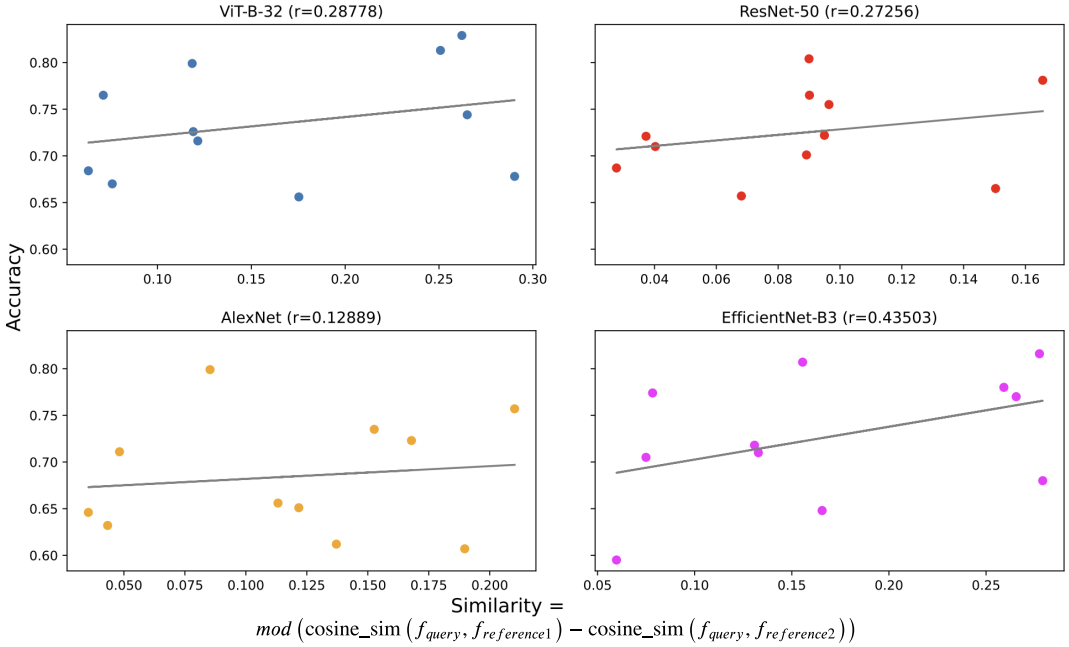

Figure 5.3: Plot of average cosine similarity difference between feature vectors generated by network for a specific triplet set and accuracy on that triplet set.

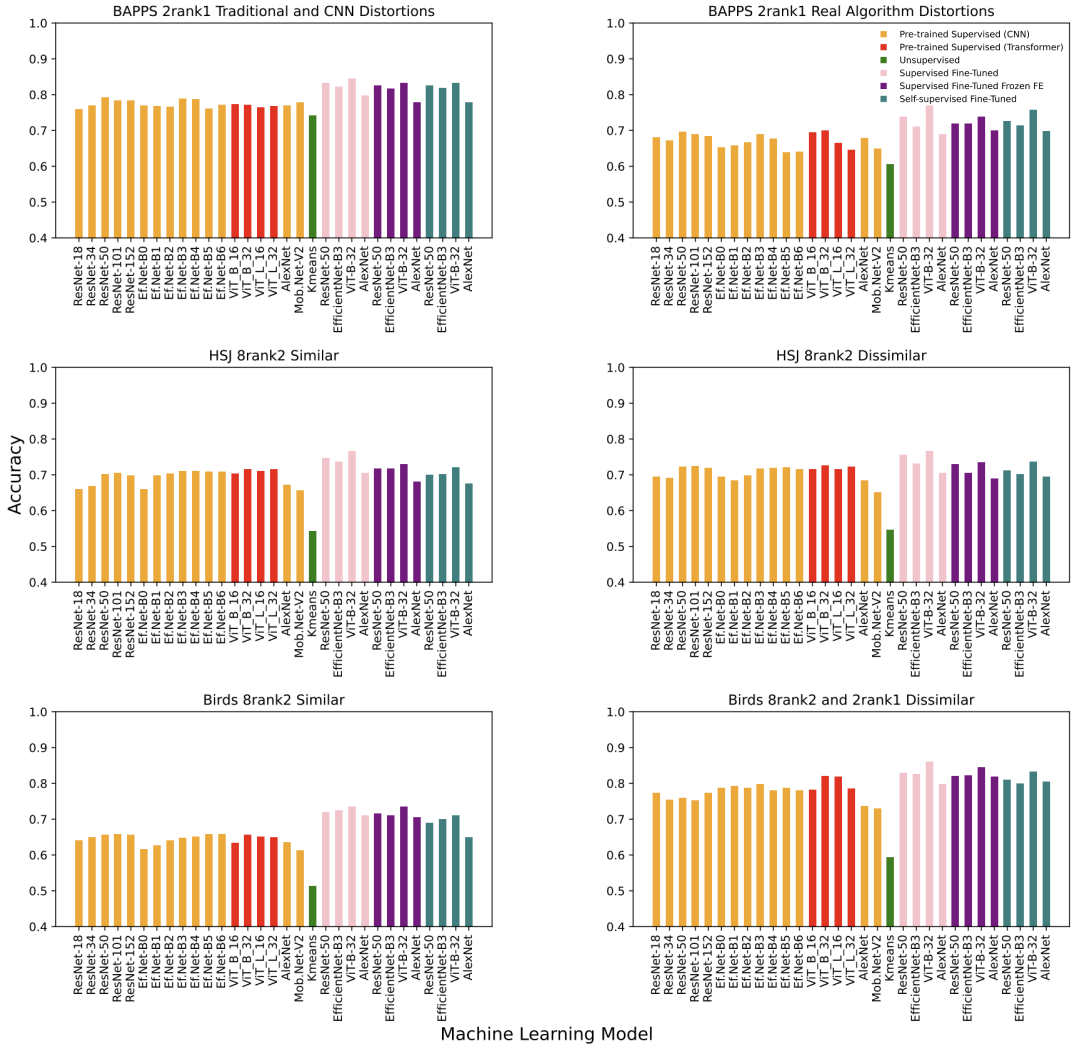

Figure 5.4: Bar chart of all results obtained following model testing and fine-tuning on each triplet set. Note: certain sets are grouped due to similarity/low informational value.

Chapter 6

Conclusion and Continuation

This chapter aims to summarize the findings discussed in Chapter 5, as well as discuss how these findings may be of use for future studies. The continuation of this project in the fifth year of study shall also be discussed.

6.1 Conclusion of Findings

Following the experiments discussed in Chapter 5, we concluded that the correlation between the untuned performance of an ImageNet classification pre-trained model on the task of predicting human similarity judgements of triplets and the model's capacity is dependent on the triplet set. More specifically, on the triplet set's similarity to ImageNet datapoints: if similar, then the correlation may be positive, if dissimilar, then this correlation may be negative. Not only, the architecture comparison discussed in Chapter 5 indicates a clear superior model at predicting triplet similarity: the Vision Transformer. This is in line with past findings discussed in Section 2.4 which affirm a closer alignment between Transformer internal representations and human perception. These findings are useful in terms of influencing the decision-making process of choosing a model architecture for the task of triplet (human judgement) similarity prediction, as well as of model complexity depending on triplet set similarity to ImageNet if the model is to be left untuned.

From a transfer learning perspective, we concluded a lack of correlation between pre-trained model performance on ImageNet classification and performance of the model's untuned features on our own triplet prediction task. This may indicate that other factors, such as model architecture or capacity, drive the difference in pe-trained model performance and shall be considered instead of transfer learning in the context of human similarity judgement prediction.

In terms of fine-tuning approaches, we concluded that supervised fine-tuning, for each architecture and model capacity, yields the best accuracy and improvement from baseline. Not only, we also conclude that higher capacity models benefit from unfrozen tuning methods, whilst lower capacity models may achieve comparable performance when tuned with their feature extraction parameters frozen.

Experiments have also attempted to understand what is that makes a task difficult for a model. Whether a dissimilarity judgement is direct or indirect did not make a difference in terms of model performance. However, after making the logical assumption that humans would find a task where the difference between the query and each reference is equal, we concluded that the same characteristic holds for models, no matter their architecture. This agrees with a past study [53] which found that, unintentionally, DNN features, irrespective of architecture, are a particularly good estimate of human similarity judgement when compared to simple pixel-level image similarity measures.

The main conclusion of the findings, however, may be the need for further development in the available datasets of human image similarity judgements - both in terms of uniqueness from ImageNet, as well as in terms of additional useful information that can be collected during these human trials - to directly link trials to the decision making process behind each similarity estimation.

6.2 Future Work: MInf Part 2

As a continuation of this project in the fifth year of study, a new dataset may be collected. As highlighted above, the human similarity judgement datasets available at the moment are either part of ImageNet or extremely dissimilar from ImageNet. ImageNet-HSJ triplets presented an advantage over triplets from other datasets when comparing quality of pre-trained model features, which did not allow for a fair comparison. On the other hand, Birds-16 presents a very niche category of judgements due to only including single bird specimens in each image. On the other hand, BAPPS revolves around a conceptually very different task, as well as a very different triplet structure which involves distortions. The new dataset to be collected should contain images obtained from scratch - newly photographed, however diverse enough to reflect a general-purpose dataset. The human similarity judgement collection process should be similar to that of ImageNet-HSJ, involving 8 rank 2 trials which can be used to generated both similar and dissimilar triplets. However, additionally, each trial should contain a way for the participant to rate the difficulty of the judgement: this will allow for a direct link to human perception that we can use to evaluate how well models align to humans from this point of view.

Another path to be explored is the use of an alternative loss functions to triplet loss. We may either modify the triplet loss function to use a variety of distance metrics or use a completely new function. For example, if we plan to collect a new dataset, we may collect it in such a way to prepare for the use of quadruplet loss (two negatives).

Lastly, we may also branch out and explore different networks from simple vision networks, such as vision-language models, which can generate descriptions of images. This leads to rich possibilities in exploring the link between human similarity judgements and the text predicted for an image. We may even use these models in the data collection process itself - perhaps randomly offering a number of participants the language model-captioned image version of a survey and a no-caption image version to the rest. We may then explore trends in whether language model captions influenced task decision difficulty or influenced the decision itself.

Bibliography

[1] Maria Attarian, Brett D Roads, and Michael C Mozer. Transforming neural network visual representations to predict human judgments of similarity. *arXiv preprint arXiv:2010.06512*, 2020.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Nicholas Baker, Hongjing Lu, Gennady Erlikhman, and Philip J Kellman. Deep convolutional networks do not classify based on global object shape. *PLoS computational biology*, 14(12):e1006613, 2018.

[4] Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Bmvc*, volume 1, page 3, 2016.

[5] Stevo Bozinovski. Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica*, 44(3), 2020.

[6] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

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

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[10] Nicole M. Gage and Bernard J. Baars. Chapter 4 - the art of seeing. In Nicole M. Gage and Bernard J. Baars, editors, *Fundamentals of Cognitive Neuroscience (Second Edition)*, pages 99–141. Academic Press, San Diego, second edition edition, 2018.

[11] Robert Geirhos, Kristof Meding, and Felix A Wichmann. Beyond accuracy: quantifying trial-by-trial behaviour of cnns and humans by measuring error consistency. *Advances in Neural Information Processing Systems*, 33:13890–13902, 2020.

[12] Robert Geirhos, Kantharaju Narayanappa, Benjamin Mitzkus, Tizian Thieringer, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Partial success in closing the gap between human and machine vision. *Advances in Neural Information Processing Systems*, 34:23885–23899, 2021.

[13] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[14] Robert Geirhos, Carlos RM Temme, Jonas Rauber, Heiko H Schütt, Matthias Bethge, and Felix A Wichmann. Generalisation in humans and deep neural networks. *Advances in neural information processing systems*, 31, 2018.

[15] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[17] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[18] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, et al. Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*, 2021.

[19] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

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

[21] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

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

[23] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962.

[24] Matt Jones and Bradley C Love. Beyond common features: The role of roles in determining similarity. *Cognitive Psychology*, 55(3):196–231, 2007.

[25] Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, pages 462–466, 1952.

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] LW Koster. Three little words–vision, perception, seeing. *Journal of Biological Photography*, 66(2):41–44, 1998.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[29] Sarah C Kucker, Larissa K Samuelson, Lynn K Perry, Hanako Yoshida, Eliana Colunga, Megan G Lorenz, and Linda B Smith. Reproducibility and a unifying explanation: Lessons from the shape bias. *Infant Behavior and Development*, 54:156–165, 2019.

[30] Manoj Kumar, Neil Houlsby, Nal Kalchbrenner, and Ekin Dogus Cubuk. Do better imagenet classifiers assess perceptual similarity better? *Transactions of Machine Learning Research*, 2022.

[31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[32] EO Lewis. The effect of practice on the perception of the müller-lyer illusion. *British journal of psychology*, 2(3):294, 1908.

[33] Grace W Lindsay. Convolutional neural networks as a model of the visual system: Past, present, and future. *Journal of cognitive neuroscience*, 33(10):2017–2031, 2021.

[34] Maya Malaviya, Ilia Sucholutsky, Kerem Oktar, and Thomas L Griffiths. Can humans do less-than-one-shot learning? *arXiv preprint arXiv:2202.04670*, 2022.

[35] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI*, pages 69–84. Springer, 2016.

[36] Dong Ping Tian et al. A review on image feature extraction and representation techniques. *International Journal of Multimedia and Ubiquitous Engineering*, 8(4):385–396, 2013.

[37] Samuel Ritter, David GT Barrett, Adam Santoro, and Matt M Botvinick. Cognitive psychology for deep neural networks: A shape bias case study. In *International conference on machine learning*, pages 2940–2949. PMLR, 2017.

[38] Brett D Roads and Bradley C Love. Enriching imagenet with human similarity judgments and psychological embeddings. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 3547–3557, 2021.

[39] Brett D Roads and Michael C Mozer. Predicting the ease of human category learning using radial basis function networks. *Neural Computation*, 33(2):376–397, 2021.

[40] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[41] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

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

[43] Marshall H Segall, Donald T Campbell, and Melville J Herskovits. The influence of culture on visual perception. 1966.

[44] Rose M Spielman, William Jenkins, Kathryn Dumper, Marilyn Lovett, and Marion Perlmutter. Psychology (openstax), 2019.

[45] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

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

[47] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[48] Shikhar Tuli, Ishita Dasgupta, Erin Grant, and Thomas L Griffiths. Are convolutional neural networks or transformers more like human vision? *arXiv preprint arXiv:2105.07197*, 2021.

[49] Amos Tversky. Features of similarity. *Psychological review*, 84(4):327, 1977.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[51] Janelle Weaver. How one-shot learning unfolds in the brain. *PLoS biology*, 13(4):e1002138, 2015.

[52] Muhamad Yani et al. Application of transfer learning using convolutional neural network method for early detection of terry's nail. In *Journal of Physics: Conference Series*, volume 1201, page 012052. IOP Publishing, 2019.

[53] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

Appendix A

Past studies

A.1 Adversary examples

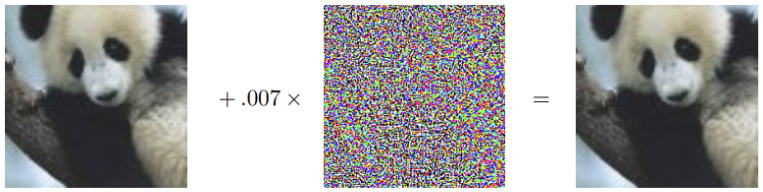

Figure A.1: *x* is the original image - whilst the original image and the perturbed right hand-side image are identical to humans, they significantly shift model predictions. Image and study credit [17].

Appendix B

Algorithms and Machine Learning Models

B.1 K-Means Visualisation

Animation credit: http://tech.nitoyon.com/en/blog/2013/11/07/k-means/

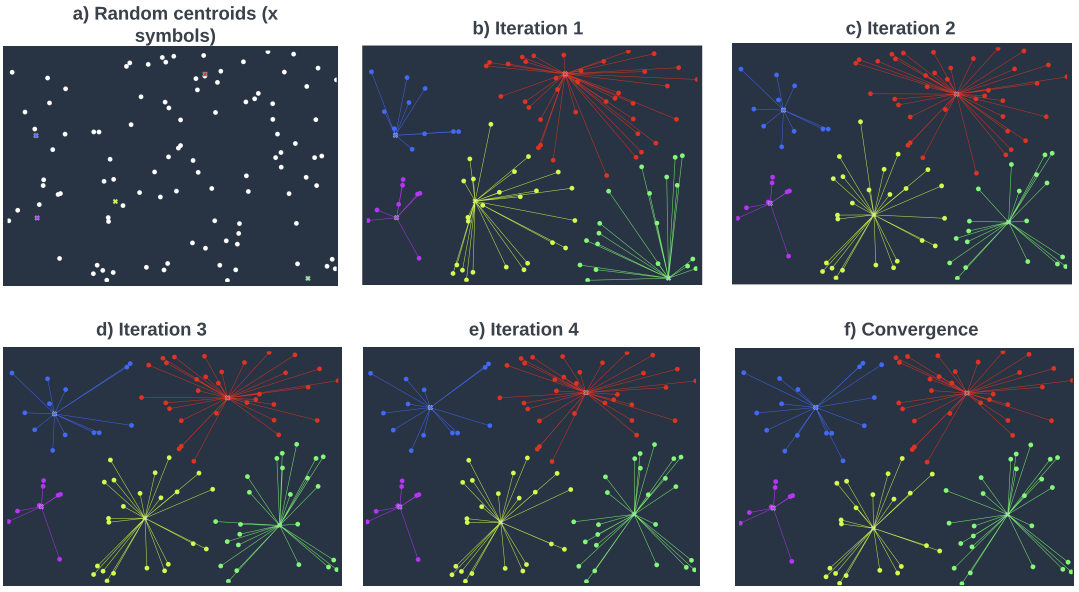

Figure B.1: A step by step visualisation of the K-Means algorithm on 50 datapoints (circles) and 5 clusters. (a) Shows the random cluster centroid (crosses) initialisation. (b-e) Show the re-allocation to clusters and the centroid update. (f) Shows the convergence of the algorithm.

B.2 Adam Optimizer Formula

The Adam optimizer equation [26].

$$
\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\
\theta_t &= \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}
\end{aligned}
\tag{B.1}
$$

B.3 Implementation Details

The implementation was achieved by:

- extracting the required triplets from the datasets of interest [38, 1, 53] using PyCharm. The 1000 triplets per dataset have been arranged in four folders per triplet set: a query image folder, a reference-1 image folder, a reference-2 image folder and a decision folder (which held .npy files indicating which reference was the most similar one). All 11 triplet sets have been zipped, uploaded to Google Drive and used on Google Colab;
- creating a Jupyter Notebook which held all implementation: deployed on Google Colab on a NVIDIA T4 Tensor Core GPU for models with under 200 million parameters and on NVIDIA V100 or A100 Tensor Core for models with over 200 million parameters.

B.4 Feature Extraction at Varying Levels

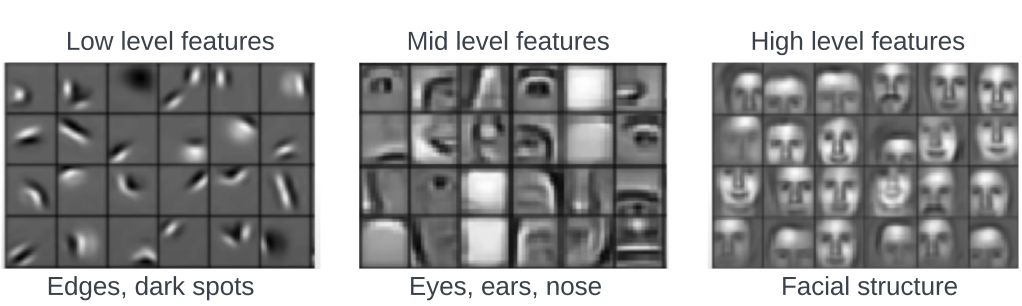

Figure B.2: A visualisation of CNN features appearing at various layers throughout the network: low level features refer to earlier layers and high level features to later layers.

Appendix C

Experiments

C.1 Parameter-Accuracy Correlation

Not included in main text due to repetition (shows the same findings as its GFLOPS counterpart).

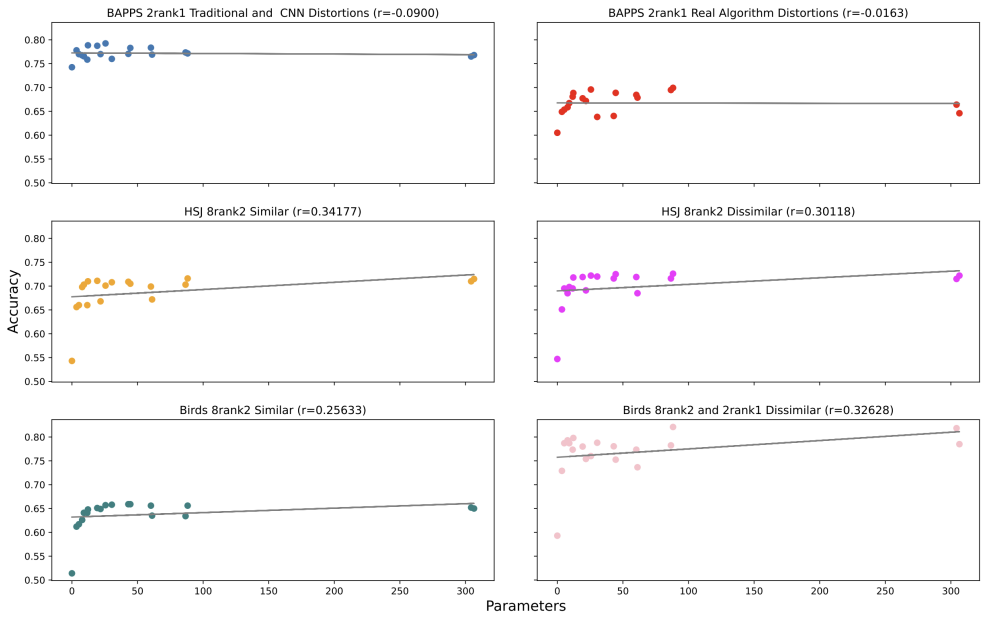

Figure C.1: Correlation plot between the number of parameters of a pre-trained model and its accuracy on the task at hand for sets of triplets.

C.2 Direct and Indirect Dissimilarity Judgement Comparison

Not included in main text due to repetition and little informational value.

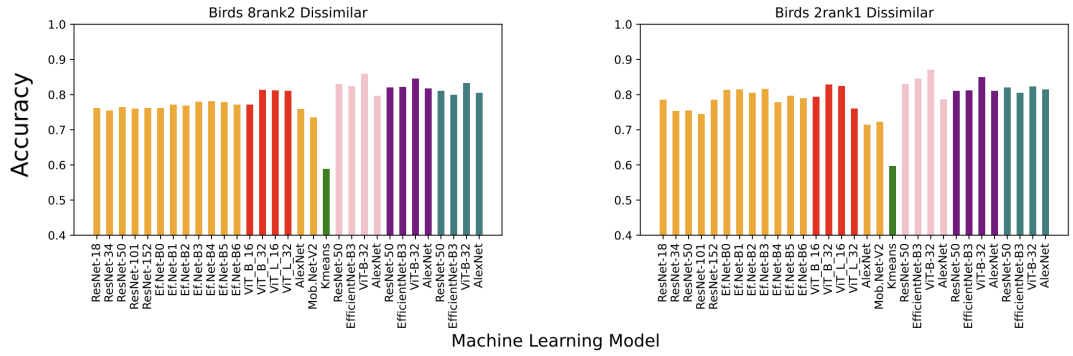

Figure C.2: Bar chart of results obtained following model testing and fine-tuning on dissimilar Birds-16 2 rank 1 triplet sets .