Personalised aesthetics assessment in photography using deep learning

Carlos Rodríguez - Pardo

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2018

Abstract

We propose a transfer-learning method capable of surpassing the state-of-the-art results in predicting personalised preferences in aesthetics in photography, while eliminating the need for data about the contents and styles of the pictures. Our methodology allows the creation of end-to-end user-specific aesthetic prediction models with a smaller set of user-specific parameters than what other transfer-learning methods would need while maintaining their predictive capability. This creates the opportunity of developing powerful content-based photography recommender systems in environments with a limited amount of computational resources.

We also show how our models could be used to understand what features make pictures aesthetically pleasant or unpleasant, as well as for the creation of personalised photography enhancements methods.

Factors such as transfer learning, data augmentation, neural network models, among other issues, are studied with the objective of obtaining a model that can accurately predict subjective preferences over pictures. Our results can be added to the corpus of the literature on this topic, and they also validate some of the results reported in other studies in machine learning, particularly about the use of residual adapters in convolutional neural networks for multi-domain learning.

Acknowledgements

I would like to thank my supervisor, Hakan Bilen, for his support, guidance and generosity throughout the development of this project. I would also like to thank my classmates for their support and the discussions in which we shared many ideas that were very valuable for this thesis.

I wish to acknowledge the help provided by all the academics and programmers who share their ideas and their open source software, making research easier and more rewarding.

Finally, I would also like to thank my friends and family for their continuous support and patience.

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.

(*Carlos Rodríguez - Pardo*)

Table of Contents

List of Figures

Chapter 1

Introduction

> Beauty in things exists merely in the mind which contemplates them, and each mind perceives a different beauty
>
> David Hume, *Of the Standard of Taste and Other Essays*

1.1 Motivation

This introduction is based on my Informatics Project Proposal report. Aesthetics perception in art is widely considered to be subjective. Different human spectators will perceive the same piece of art differently, which hinders the task of identifying which features make a piece of art aesthetically pleasant or unpleasant. Cognitive studies suggest that the perception of beauty in humans is a multi-step process in which several factors are involved. Among them, some can be highlighted, such as the objective features of the piece of art itself (contrast, colours or symmetry, style, etc.), the situation surrounding the perceiver (social interactions, mood, etc.) and the perceiver's past experiences (personal tastes and biases, preferences, knowledge, etc.) (Leder et al., 2010).

The fact that the perception of beauty depends on so many factors, in ways that are unique for each person, and which are not fully understood, can suggest that it is not possible to assign an objective score of beauty to a piece of art. Nevertheless, some studies indicate that there is indeed a shared sense of beauty among humans (Hayn-Leichsenring et al., 2017), which can be used to approximate a non-subjective set of

features that can distinguish between beautiful and unpleasant pieces of art.

Despite early scepticism, photography is now considered an art form (Adams, 1982) on the same level as music, theatre or paintings. However, the wide availability of digital cameras and the impact of visual culture has made photography one of the most popular art forms. Social networks have contributed to this ubiquity of photography in everyday life. They are a form of communication, journalism and social interaction. The same picture can be seen at the same time by thousands of people in different places. This reproducibility makes photography somewhat different to other forms of art like live music, dance, paintings or theatre. In Walter Benjamin's words (Benjamin, 1935):

> In even the most perfect reproduction, one thing is lacking: the here and now of the work of art, its unique existence in a particular place. It is this unique existence and nothing else that bears the mark of the history to which the work has been subject.

Even with this difference, as any other art form, photography is not immune to individual preferences and cultural tendencies. Moreover, the fact that the same picture can be seen in different media (printed, electronic devices with different resolutions, colour depths, etc.) and situations can provoke that the perception of beauty in photography is more subjective than other forms of art. Nevertheless, professional photographers have suggested a list of features that can indicate whether a picture is beautiful or not. Among those, symmetry, composition, contrast, perspective, colour palette, zoom or illumination stand out (Sheppard and Martin, 2009; Datta et al., 2006; Obrador et al., 2010). Despite photographers' efforts on listing such features, it is not possible to directly predict the aesthetic value of a picture using a combination of those features, which suggests that there are other features (or combinations thereof) that are related to the perception of beauty in photography that have not yet been identified due to their abstract nature or humans' lack of ability to understand the processes that control their preferences. In fact, some renowned photographers argue that those features do not exist: "There are no rules for good photographs, there are only good photographs" — Ansel Adams (Schoch, 2001). Assuming these features do exist, the set of features that are important for one spectator may not be as important for another spectator, which can make the identification of said features even harder.

Machine learning and other computational methods can be used to overcome some of these difficulties. Instead of theorising about what features can make a picture more beautiful than others, machine learning makes it possible to learn those features from

data. In fact, machine learning has been used many times in the past to study different problems regarding aesthetics in photography. In most of those studies, the solution to the problems involved using a dataset of pictures and their corresponding aesthetics value (a numeric rating or a categorical rating), which was usually computed as the mean rating that several spectators gave to that picture. Some of the datasets made use of additional annotated information (e.g., information of the contents of the pictures or information about style, composition, etc.) to help researchers solve their tasks. Some authors have used this kind of annotated data to find the relation between some characteristics in photographs and their aesthetic value (Appu Shaji, 2016; Lu et al., 2014; Deng et al., 2017). Other authors argue that the popularity of a picture and its aesthetic value are not necessarily related, and propose methods to predict the number of times a picture would be seen by the users of a photography-centric website (Khosla et al., 2014).

The studies that have been mentioned do not take into account individual preferences and subjective factors, and, in consequence, they were not able to predict the rating that a particular spectator would give to a picture that they had not seen before. We can find studies that do take into account these factors, like (Park et al., 2017), which combine a support vector regression (SVR) model with a ranking support vector machine (SVM) to predict individual preferences using interaction data.

In (Ren et al., 2017), a convolutional neural network (CNN) is combined with a SVR with a radial basis function for predicting how a known user will rate an unseen picture. The CNN is used to predict an expected mean rating of a given picture, and the SVR computes the offset from that rating, given the past preferences of the user. In other words, the CNN predicts how an average user would rate a picture and the SVR is used to compute how different from the mean rating the individual rating would be. The SVR is trained individually for each user, using information about the style and contents of the pictures. They propose two different datasets with unique characteristics which facilitate the research on individual preferences on image aesthetics by not only providing the average rating of each picture (which is a common practice in this area of research), but also providing information about how each user rated each picture.

1.2 Objectives

The main goal of this thesis is to build on Ren et al.'s work, and, by using the same dataset they used and provided, design and test a single end-to-end machine learning model that can take into account individual preferences in the aesthetics prediction on photography. One of the main goals will be to overcome Ren et al.'s need for information about the styles and contents of the pictures, which can be expensive to obtain and limiting.

Several models will be studied. The goal of every model will be to, given only the ID of a user and the pixels of a picture, predict the rating that said user would give to that picture, without using any additional information. The process will be as follows: First, a baseline model will be trained so that it can predict the mean rating that a set of users would give to a picture. From there, different models will be tested, so that they can learn, using limited data, how new users would rate new pictures. These models will use information (features) that is embedded in the baseline model. There is an implicit assumption in this process, which is that the features that explained the preferences of the users whose ratings were used to train the baseline model will be similar to the preferences of new users. By embedding user preferences in the models, it is possible to find patterns and features that explain a shared sense of aesthetics in photography, as well as individual preferences, as the models will be learning both kinds of preferences.

The proposed models will be different versions of convolutional neural networks, which have the power of learning meaningful features in pictures, as will be explained later in this report. The first experiment will involve fine-tuning the weights of the last layers of the baseline network so that the feature maps of the early layers of the network stay fixed. After that is done, fine-tuning will be performed on every layer of the network. Finally, some experiments will be built on (Rebuffi et al., 2017)'s suggested transfer-learning models. This involves adding small adapters throughout the network, so that individual preferences with different levels of abstraction can be learned. These experiments will be detailed later in this thesis. In addition to eliminating the need for manually annotated data, the models could also improve the performance reported in (Ren et al., 2017), because the individual preferences will be learned using only convolutional neural networks, which can have more predicting power than support vector regressors.

Several machine learning problems are involved in this project. Transfer learning,

data augmentation, network configurations and hyperparameters, residual connections and adapters, cross-validation and other topics have an important role in the findings of this project, which is why they will be thoroughly explained in subsequent sections of this document.

1.3 Thesis outline

The rest of this thesis will be divided into several sections. Firstly, some background about the use of machine learning in photography will be given. Topics regarding machine learning for image recognition will be explained, such as convolutional neural networks and topics related to them, to provide a theoretical and empirical justification for the experiments performed in this project. Additionally, the state-of-the-art methods on image aesthetics predictions will be outlined by studying the literature on the topic more deeply that it was done in the introduction of this document.

Secondly, building upon that literature review, design issues will be explained, such as the decision on what kind of network architecture to use, as well as ways of evaluating the performance of the models. From there, implementation issues will be outlined, such as which deep learning library was used and why, which kind of training was used, data preprocessing, etc. A more detailed description of the dataset that was used to train and evaluate the models will also be provided.

From there, the experiments performed throughout this project will be explained and their objectives will be justified. The results of those experiments will be summarised in the following chapter.

Finally, building on the results of the experiments, some conclusions will be drawn with a particular focus on the goals of the project. The impact of this project on the literature will be outlined and some future work will be proposed so that the findings of this project can be put into a wider context.

Chapter 2

Background

> There are always two people in every picture: the photographer and the viewer.
>
> Ansel Adams

2.1 Image classification

Image classification is one of the most studied problems in the Machine Learning and Computer Vision literature (201, 2018). It has a wide variety of applications, including autonomous driving, optical character recognition or face recognition (Li Deng, 2012; Lawrence et al., 1997; Chen et al., 2016). It is the task of predicting to what category from a fixed set of labels an input image belongs to.

The Machine Learning approach to solve this problem is data-driven. Typically, a learning algorithm will learn features from a set of examples known as the training set, and evaluate its performance on examples that the algorithm has not been trained on, which composes the test set. Additionally, it is common to use a validation set to tune the hyper-parameters of the learning algorithm, with the hypothesis that, if a change in a hyper-parameter leads to a better performance on the validation set, it will probably lead to a better performance on the test set. This is because, ideally, the validation and training sets should follow the same distribution.

Images are commonly represented in computers as 3-dimensional rectangular matrices of pixels, which represent the intensity of a colour in each position of the image. Depending on the format, images can be represented using 3 dimensions of colours

(red, green and blue in the RGB format), 1 dimension for grayscale images or 4 dimensions for images with transparency (PNG format) (201, 2016). Different image formats have different colour depths (bit-depth), which relates to the range of intensities that they can represent.

The way in which images are represented in computers is important for image classification algorithms, because the inputs for a learning must be represented in the same way. Nevertheless, state-of-the-art image classification algorithms use tensors to represent images, and it is possible to transform most raster image formats (.png, .jpg, etc.) into tensors. The conversion between some image formats can be lossy, as some quality is lost, but the impact of such loss in the performance of image classification algorithms has been shown to be limited (Dodge and Karam, 2016).

2.1.1 Challenges in image classification

One of the main tasks of image classification is identifying what object is in a picture. There are several particularities in images that make this task more difficult. Objects can be rotated and scaled (rotation and scale variation), deformed, illuminated in different ways, occluded, etc. (201, 2018). Machine learning algorithms may need to learn features that do not depend on those factors. Moreover, the labels in which the pictures are classified into can be very broad, which adds additional complexity to the problem.

A common problem in machine learning is the lack of training data. This problem happens when the amount of data that can be used to train the learning algorithm is too small for the algorithm to learn meaningful features with. Fortunately, there are techniques that can be used to solve some of the challenges commented before and the lack of data. Those techniques are know as data augmentation techniques, and are widely used in the image classification literature. They perform random transformations on the images in the training set, so that the algorithm can learn more generalizable features. Those transformation include rescaling the images, randomly cropping them, flipping them horizontally or vertically, adding random noise or rotating them. Recently, other methods have been proposed, including the use of generative adversarial networks (GANs) (Radford et al., 2016; Raj, 2018; Wang and Perez, 2017). Depending on the problem, the use of each of those techniques can be beneficial or detrimental in the performance of the algorithm, and, consequently, they must be chosen carefully.

There are several image classification algorithms that have been used in the past

years. Traditionally, the algorithms with the best performances were versions of discriminative algorithms like Support Vector Machines (SVM) in addition with methods like the Fisher Vector (Sanchez et al., 2013). Nevertheless, the parallel increase in the amount of available data, the computational power of electronic devices and the development of new machine learning algorithms have lead to the popularization of neural networks, which, as of today, are the state-of-the-art set of algorithms in the image classification literature.

2.1.2 Neural Networks

Neural networks are machine learning models vaguely based on biological networks of neurons. They can solve numerous problems, from supervised learning (classification or regression) to unsupervised learning, as in auto-encoders or self-organising maps (Owens and Hunter, 2000; Lange and Riedmiller, 2010). In many applications, the state-of-the-art model is some variation of a neural network.

In supervised learning settings, neural networks are usually defined as a directed sequence of layers, in which a data example is fed into the first layer of the network, which performs a mathematical operation to it, and feeds the result of that operation into the next layer. This is the case of *multilayer perceptrons* (MLPs), which are the most common neural network model. Those models' goal is to find an approximation f to an unknown function f^* that maps an input x to an output y, so that $f^*(x) = y$. This function, in MLPs, depends on the input and a set of parameters θ. Learning in MLPs is finding the value of θ such that $y = f(x;\theta)$ is the best possible approximation to f^* (Goodfellow et al., 2016). Those models are also known as *Deep Feed-forward Networks*, which are a subset of the models known as deep learning models (LeCun et al., 2015).

In order to learn the value of the parameters, the network receives inputs (an example from a training dataset), computes the output, and compares it to the real value (class or number assigned to that example). The difference (defined using a loss function) between the network's output and the real value is back-propagated from the last layer to the first layer of the network using some optimization method, such as gradient descent. The parameters, or weights, of the network, are updated in a way that, if the same input is shown to the network again, the error it will make should be smaller than before the weight update.

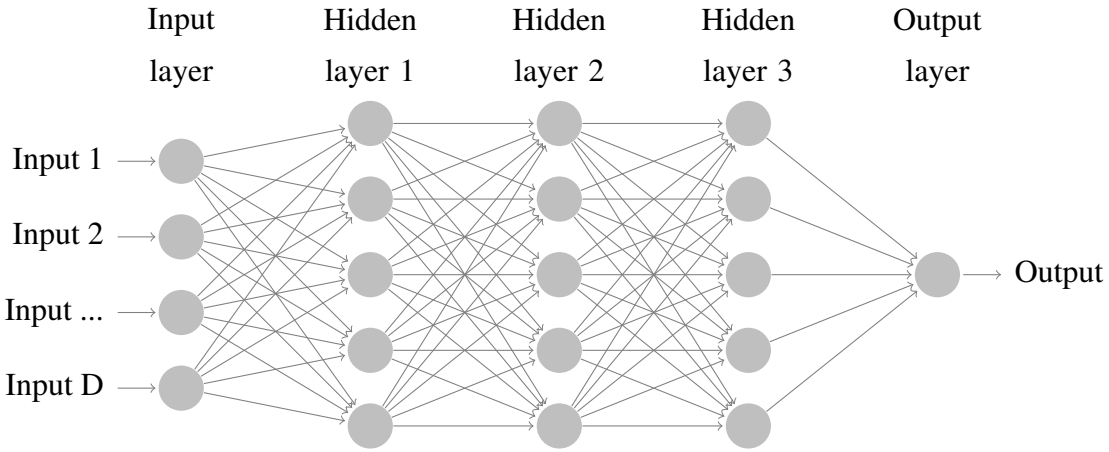


Figure 2.1: Diagram of a sample neural network with 3 hidden layers with 5 hidden units each.

One of the most fundamental parts of MLPs are hidden layers. They are layers that receive an activation from a previous layer, perform a set of operations on them, and propagate their own activations $x_1, x_2, ..., x_D$ to the next layer. The operations are performed by hidden units, or neurons, that receive a weighted linear combination of the activations of the previous layer $(W^T x + b)$, and output a single scalar value y that depends on said combination of activations $g((W^T x + b))$. The value of b is also learned during the training process. The operation g is known as the activation function. Each hidden layer has a set of hidden units with no connections among them (thus feed-forward neural networks), each of which receives a different combination of activations, as the weights assigned to each hidden unit is different.

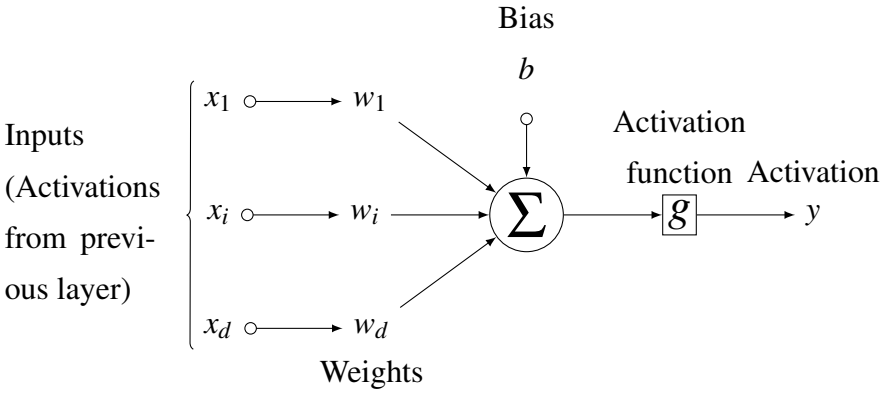


Figure 2.2: Diagram of an activation function in a neural network

Activation functions

There are several activation functions that have been used throughout the MLP literature. According to the *Universal Approximation Theorem*, any feed-forward network with one single hidden layer with enough hidden units can approximate any function if the activation functions of the hidden units are continuous and monotonically increasing functions (Cybenkot, 1989; Leshno et al., 1993). It was previously believed that the functions also needed to be bounded, but this has been proven wrong recently (Sonoda and Murata, 2015). In order to be able to use optimization methods like gradient descent, this function must also be differentiable.

Sigmoidal functions Hereunder, some of the most common activations functions will be explained. It will be assumed that z is the weighted combination of the inputs of a hidden unit $z = W^T x + b$. One of the most common starting points when choosing an activation function is to choose the sigmoid function $\sigma(z)$ (also known as logistic) or the hyperbolic tangent function $tanH(z)$. Those functions have similar properties, and they are closely related, as $tanH(z) = 2\sigma(2z) - 1$.They are defined as follows:

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{2.1}$$

$$tanH(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.2}$$

Sigmoidal functions have properties that can make their use attractive. For example, they can be used to predict the probability of one event given the input data, because they are bounded between 0 and 1. Nevertheless, computing their output involves computing exponentials and divisions, and they saturate when z is either very high or very low. This is problematic for optimization methods like gradient descent, which motivates the use of different types of activation functions (Goodfellow et al., 2016).

ReLU One of the most popular alternatives to sigmoidal activation functions is the Rectified Linear Unit, commonly referred to as ReLU. ReLU has advantages with respect to sigmoidal activations. First, it is computationally less expensive to operate with, as there are no exponentials or divisions, its derivative is always either 0 or 1 (which is an additional computation advantage) and its output is independent to the scale of the input, as $max(0, ax) = amax(0, x)$. It is defined as:

$$Relu(z) = max(0, z) \tag{2.3}$$

Despite its advantages with respect to sigmoidal functions, ReLU has problems when using gradient descent as an optimization methods. Its derivative is equal to 0 when $z < 0$, which prevents back-propagation of the error in ReLU units whose activations are negative. This is known as a *dying ReLU* Liu (2017). To overcome those disadvantages, several modifications to ReLU have been proposed.

PReLU and Leaky ReLU A simple solution to the dying ReLU problem is to change the activation to:

$$PReLU(z) = \begin{cases} \alpha \times z & \text{if } z \leq 0 \\ z & \text{if } z > 0. \end{cases} \tag{2.4}$$

The difference between PReLU (Parametric rectified linear unit) and Leaky ReLU is the value of α. In Leaky ReLU, the value of α is smaller (e.g. $\alpha \leq 0.01$). In PReLU, the value of α can be 0.25. There are issues with both PReLu and Leaky ReLU that can justify the use of other activation functions, such as SELU (Scaled Exponential Linear Unit), whose mean activation is 0, or ReLU-6 (Clevert et al., 2016; Krizhevsky, 2010). Nevertheless, ReLU and its variations work in practice, and are part of models that have achieved state-of-the-art results in several problems (Targ et al., 2016; Szegedy et al., 2016b).

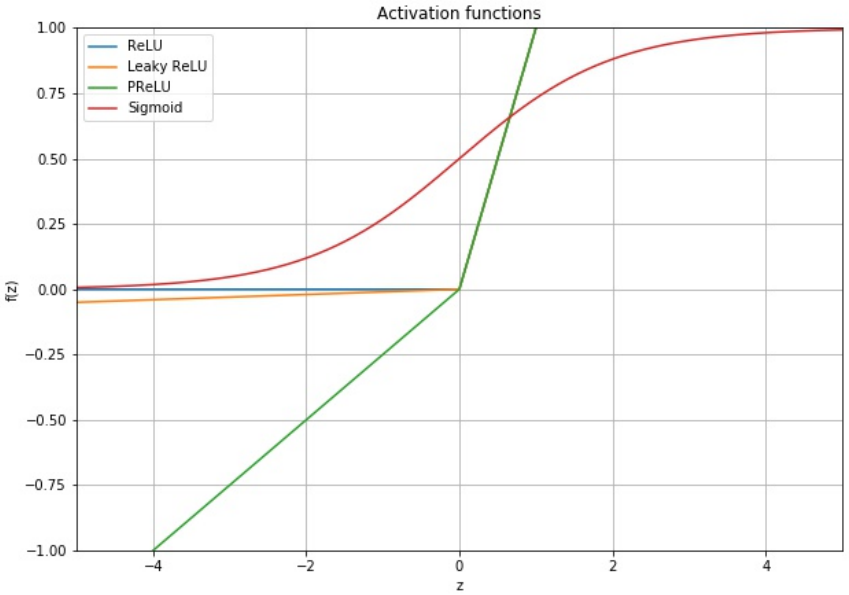


Figure 2.3: Visualization of some activation functions

Loss functions

Another important factor to take into account when solving a classification or regression problem using a MLP is the choice of a loss function. A loss function J s a mathematically function that computes the difference between the network's output $\hat{y} = f(\theta, x)$ given the input x and the target output y. The network will learn to minimize the loss function given the training dataset by modifying θ using an optimization method. That is why the form of $J(\theta; x, y)$ is relevant, as different loss functions have different properties, and, depending on the problem, one loss function may be more appropriate than other. When using a gradient-based optimization method, the network will update its parameters by taking the partial derivative of the loss function with respect to the parameters: $\theta^{i+1} = \theta^i - \frac{\delta J(\theta)}{\delta \theta^i}$. This requires the loss function to be differentiable with respect to the parameters (Agrawal, 2017).

In regression problems, where the target value is a real-valued number, one of the most common loss functions is the mean squared error (MSE), which is defined as (for N data points):

$$MSE(\theta; x, y) = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(\theta, x_i))^2 \tag{2.5}$$

MSE is one of the most frequently used loss functions in regression problems, but there are others, like the Mean Absolute Error ($MAE(\theta; x, y) = \frac{1}{N} \sum_{i=1}^{N} (y - f(\theta, x))$. MSE, as it takes the square difference between output and target, heavily penalises outliers (Varma, 2018).

For classification, one of the most popular loss functions is the cross-entropy loss. This loss assumes that the input x can be assigned to a number k of classes, and that the output y is defined as a one-hot vector, in which each position y^i of the vector is the probability of x being classified as the label assigned to i. The cross-entropy error is defined as follows (for N data points):

$$CSE(\theta; x, y) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k}^{K} -(y_i^k log(f(\theta, x_i)^k) \tag{2.6}$$

Those two functions are common in the MLP literature, but there are others, like the Hinge Loss, Softmargin Loss, Binary Cross-Entropy, Kullback-Leibler divergence Loss, etc. Nevertheless, for image classification or regression, cross-entropy loss and MSE are two of the most common cost functions. The nature of the problem is important when choosing a loss function, but this choice can also have impact in the performance of the network, training times, etc. (Janocha and Czarnecki, 2017).

Optimization

Optimization in MLPs refers to the process of adjusting the weights of the network so that the loss function is minimized. There are several methods available to perform this task, but the most commonly used are gradient descent methods. Second-order methods like Newton's method will not be covered in this section, as they are harder to compute in terms of time and memory.

Gradient descent As was pointed out before, gradient descent methods update the weights of the parameters of the network by taking the first partial derivative of the loss function with respect to the weights of the network $\theta^{i+1} = \theta^i - \eta \frac{\delta J(\theta)}{\delta \theta^i}$. The parameter η is the learning rate, and controls how much each parameter is updated at each iteration of the algorithm. The way of propagating the loss from the output layer to earlier layers of the network is called back-propagation of error, which is the algorithm that allows the training of MLPs (Rumelhart et al., 1986). Gradient descent is guaranteed to reach a global minimum on the error surface for convex surfaces, and to a local minimum for non-convex surfaces (Ruder, 2016). Convex error surfaces are infrequent in many real problems, and getting stuck in a local minimum leads to suboptimal solutions that may not satisfactory. The value of the learning rate controls a trade-off between stability (if η is too big, the algorithm may not converge) and training speed (if η is too small, the algorithm may take a long time to reach a minimum).

Baseline (batch) gradient descent needs to go through all the training set, compute the loss for each example in the set, and average the weight updates over all losses. This has several disadvantages, as it requires a significant amount of memory, and can be very slow. That is why some modifications have been used in the MLP literature.

Stochastic gradient descent Instead of going through all the training set before each weight update, stochastic gradient descent computes the loss for one data example and updates the weights using only the loss of that example. This is done for each example in the training set. Each weight update is an approximation to the real gradient, which can be an advantage as it can avoid getting stuck in local minima, but learning can also fluctuate in an unacceptable way.

Mini-batch gradient descent A middle ground between batch gradient descent and stochastic gradient descent is called mini-batch gradient descent. It involves taking a fixed number (> 1) of examples from the training set, computing their loss and

averaging the weight updates of all the examples in that mini-batch. It can compute more accurate weight updates and it can be more memory-efficient than stochastic gradient descent, without the disadvantages of batch gradient descent. It adds some extra complexity, as the size of the mini-batch has to be chosen manually.

Learning rules

So far, it has been argued that the weight update for each weight depends on the loss function and a fixed η which is shared for all weights of all layers. This has issues, like the possibility of getting stuck in suboptimal minima or weight updates instability. Several modifications to the baseline learning rule have been proposed. Hereunder, some of them are listed and briefly explained.

Momentum One way of adding stability to the weight updates, particularly in stochastic gradient descent learning, is to add a *momentum* parameter to each weight update. This parameter makes the updates depend on the current loss and the direction of previous updates, which results in more stable updates. The momentum weight update rule is (Goodfellow et al., 2016):

$$v \leftarrow \alpha v + \eta \nabla_\theta J(\theta; x, y) \tag{2.7}$$

$$\theta \leftarrow \theta + v \tag{2.8}$$

A variation to the standard momentum learning rule is the Nesterov momentum. In Nesterov momentum (Sutskever et al., 2013), the gradient is computed after the current momentum is added to the network's weights:

$$\theta' \leftarrow \theta + v \tag{2.9}$$

$$g \leftarrow \nabla_{\theta'} J(\theta'; x, y) \tag{2.10}$$

$$v \leftarrow \alpha v - \eta g \tag{2.11}$$

$$\theta \leftarrow \theta + v \tag{2.12}$$

$$\tag{2.13}$$

Both standard momentum and Nesterov momentum require a manual setting of the hyper-parameters α and η, which can have a significant effect on the performance of the network. They are constant hyper-parameters for each weight of the network, which has theoretical disadvantages (Goodfellow et al., 2016). One way to overcome those difficulties is to have adaptive learning rates of the model parameters.

Learning rules with adaptive learning rates There are several methods that can adapt the learning rate of the parameters of the network, like Adagrad or Adadelta (Zeiler, 2012). Two of them stand out from the others due to their capabilities and their empirical success.

RMSProp RMSProp (Hinton et al., 2012) is a modification of the Adagrad algorithm that adapts the learning rate of each parameter by computing an exponentially weighted moving average of the gradient. Given a learning rate η, a decay rate ρ and a small constant ε (e.g, 10^{-7}), RMS prop computes the gradient updates as follows:

$$g \leftarrow \nabla_\theta J(\theta; x, y) \tag{2.14}$$

$$v \leftarrow \rho v - (1-\rho) g^2 \tag{2.15}$$

$$\theta \leftarrow \theta - \frac{\eta}{\sqrt[2]{v+\varepsilon}} \times g \tag{2.16}$$

The reason behind using the exponentially moving average is to give more relevance to recent weight updates. RMSprop, despite its advantages, does not take into account the momentum of the weights updates.

Adam Adam (Kingma and Lei Ba, 2014) is a learning rule that combines Momentum and RMSprop, in the following way. Given a step size η, a small constant ε (e.g, 10^{-7}) and two exponential decay rates ρ_1 and ρ_2, with suggested values of 0.9 and 0.999, the weight updates are done following the next steps:

$$g \leftarrow \nabla_\theta J(\theta; x, y) \tag{2.17}$$

$$t \leftarrow t + 1 \tag{2.18}$$

$$s \leftarrow \rho_1 s + (1-\rho_1) g \tag{2.19}$$

$$r \leftarrow \rho_2 r + (1-\rho_2) g^2 \tag{2.20}$$

$$\hat{s} \leftarrow \frac{s}{1-p_1^t} \tag{2.21}$$

$$\hat{r} \leftarrow \frac{r}{1-p_r^t} \tag{2.22}$$

$$\theta \leftarrow \theta - \eta \frac{\hat{s}}{\sqrt[2]{\hat{r}}+\varepsilon} \tag{2.23}$$

Unlike RMSProp, Adam has biases compensation methods, and has been shown to be robust to the initial choice of hyper-parameters (Goodfellow et al., 2016). Adam can also incorporate Nesterov Momentum updates.

Batch Normalization

Batch Normalization (BN) (Ioffe and Szegedy, 2015) is a technique developed to solve some problems faced when training deep neural networks. It can allegedly considerably reduce training times by accelerating convergence, it allows to use higher learning rates, can improve the predictive performance of the network and reduce the need for regularization methods in the network. It works by normalising the activations of each hidden layer in the network with the goal of reducing *Internal Covariate Shift*, which is the "change in the distribution of network activations due to the change in network parameters during training" (Ioffe and Szegedy, 2015). After normalising the activations, BN performs an additional operation of scaling and shifting $y = \gamma x + \beta$, where γ and β are learned during training.

Batch Normalization has become an important part of very deep neural networks, and they are, by default, incorporated in state-of-the-art models such as Resnet or Inception (He et al., 2015; Jin et al., 2016). Some improvements on BN have been proposed, as Layer Normalization (Ba et al., 2016) .

Regularization

Regularization methods are techniques applied to machine learning models with the goal of improving their generalization error, by avoiding over-fitting (fitting noise in the training data). Some of the most common regularization methods involve constraining the value of the parameters of the model, thus avoiding them to become too big. Among those, L^2 (also known as weight decay) and L^1 stand out as two of the most typical ways to perform regularization. They involve the reduction of the absolute value of the weights of the network before each weight update, thus controlling their growth (Goodfellow et al., 2016).

Dropout Recently, a simpler method was proposed. Dropout (Srivastava et al., 2014) works by deactivating some activations during training time, which prevents the co-adaptation of feature extractors. It has shown success in several computer vision problems, and has become one of the most widely used regularization methods, thanks to its simplicity and performance. The need for the use of several regularization methods within the same network has been decreased by Batch Normalization. BN, while it is not a regularization method, it can solve some of the problems that other regularization methods solve.

Convolutional Neural Networks

So far, it has been assumed that the layers in the MLP were fully connected layers with standard activation functions. This kind of architecture, while powerful, is limited in domains in which the data has patterns that depend on local structures. Audio and images are examples of such domains. A pixel in an image is more related to pixels close to it than to other pixels. Convolutional neural networks (CNNs) are a special kind of deep feedforward network that use a particular kind of layer, a Convolution layer, along with other particularities. They were inspired by the cat's visual cortex (Hubel and Wiesel, 1962), and were first presented in (LeCun et al., 1998a), but they did not show significant success until AlexNet (Krizhevsky et al., 2012). The idea behind them is to train a set of feature extractors (the convolutional layers) on top of a baseline fully connected classifier, as presented in (LeCun et al., 1998a).

Convolutional layers The main component of CNNs are convolutional layers, which are able to learn features that exploit local patterns, in a similar way as receptive fields on mammals' visual cortex do. Convolutional layers are a set of filters or kernels (in image recognition, they are typically 2-Dimensional square filters). The convolution operation involves sliding each filter across the input image (or the activations of the previous layer), thus obtaining *feature maps*. Intuitively, each feature map indicates the presence or absence of a particular visual feature (corner, edge, texture, etc.) in a region of the input picture. When the network is trained, it learns by gradient descent those features that can better separate classes, in the case of classification problems. One of the advantages of CNNs is that, when multiple convolutional layers are concatenated, they learn features with hierarchical levels of abstraction. Earlier layers will look for simpler features like edges, while deeper layers would look for more abstract characteristics, like textures.

In convolutional layers, there are four main parameters that can be tuned (201, 2018).

Filter size This parameter controls the dimension of the features that will be searched in the image. The bigger the parameters size, the more weights that will be optimized, adding complexity to the model, but it can lead to finding more relevant features as more pixels are involved. Some common filter sizes are 5x5, 3x3 or 7x7.

Depth of the layer Convolutional layers have depth on their own. This corresponds to the number of filters of each layer. Typically, in deep CNNs, the deeper the convolutional layer, the more filters it will have.

Stride This parameter controls how many times each filter is convolved through the input image. If the stride is set to be equal to 1, the filter will be slid one pixel at a time, whereas if its equal to 2, it will ignore one of every 2 pixels, thus reducing the dimension of the output.

Padding Padding is an operation that adds zeros around the border of the image, so that features can be looked for in the limits of the image. A padding of (1,1) will add one zero next to each of the pixels in the border of the image. This parameter also controls the size of the output of the layer.

There are some other parameters, like dilation, but they are not regularly used.

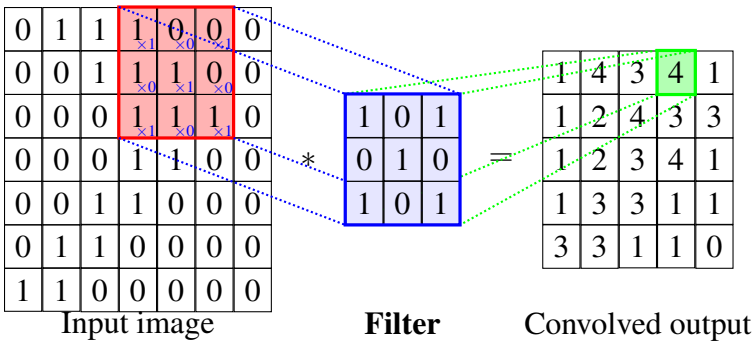


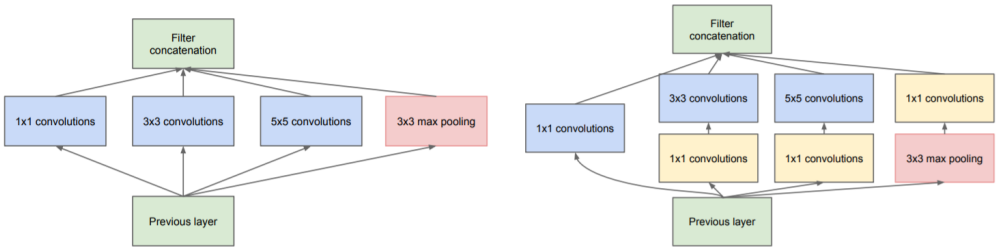
Figure 2.4: Example of a 2-dimensional convolution. The filter size is 3x3, and it uses a 0-padding configuration. Figured based on image from `https://github.com/PetarV-/TikZ/tree/master/`

Pooling layers In some deep CNN architectures, it is common to find pooling layers after successive convolutional layers. They work in a similar fashion as convolutional layers do, but instead of returning the result of a convolution between the input image and a filter, they return the maximal activation (in the case of max-pooling layers) or the average of the activations (in the case of average-pooling layers) from a pooling window (typically 2x2). Their goal is to reduce the size of the outputs of hidden convolutional layers, thus reducing the number of parameters in the network and reducing the probability of over-fitting.

Other layers In addition to convolutional and pooling layers, deep CNN can also add a batch normalization operation right after each convolutional layer, and a non-linear activation (commonly ReLU) after the batch normalization *layer*. Sometimes, when referring to *convolutional layer*, authors are referring to a layer composed of a convolutional stage, a normalization stage, a non-linearity and a pooling stage (Goodfellow et al., 2016).

State-of-the-art convolutional neural network architectures Convolutional neural networks are the state-of-the-art family of models in many image recognition tasks. In the machine learning literature, there are some datasets that are used to compare the performance of different models. In image recognition, some stand out, as MNIST (a collection of grayscale handwritten digits), Cifar-10 or ImageNet (LeCun et al., 1998b; Krizhevsky, 2009; Ima). Among the top-performers on the ImageNet challenge, three network architectures stand out.

Inception Inception CNN architectures (Szegedy et al., 2014, 2015) (also known as GoogleNet) are a family of deep CNN that make use of Inception Modules. These modules are a set of convolutional layers with different filter sizes computed in parallel, and whose outputs are concatenated. This allows to find, from the same input image, different features with different sizes, and reduces the need of pre-determining which filter size is better, as many sizes are trained in parallel, and the network will learn to weight them. A concatenation of 9 Inception Modules defines the whole Inception V1 architecture, and many improvements have been made to this popular architecture through the recent years.



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2.5: Inception Module. Obtained from (Szegedy et al., 2014)

VGGNet In (Simonyan and Zisserman, 2015), the popular VGGnet was presented. Several configurations of it were presented, but the most popular is the VGGNet with 16 layers, as it has less parameters and better generalization results than the VGGNet with 19 layers. In contrast to Inception networks, all the filters in the VGG are of the same size, (3×3). The VGGNet 16 has 13 convolutional layers with increasing number of filters when going deeper in the network, and a bottleneck with 3 Fully Connected layers with ReLU activation functions.

Resnet Residual Networks (He et al., 2015), commonly referred to as ResNet, are a family of CNNs that have shown considerable success in image recognition tasks. The main difference between residual networks and other CNNs is the use of residual connections, which allow the training of networks with many layers (up to 8 times more depth than a VGG 16) but with less parameters. A Resnet-152 (with 152 layers) outperforms VGGnet16 and Inception V1 in classification accuracy on the ImageNet challenge.

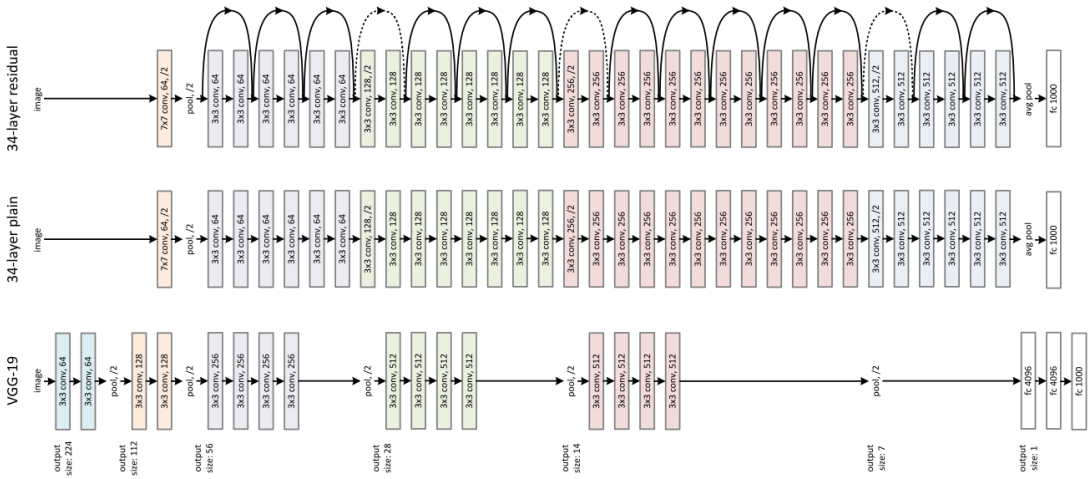


Figure 2.6: From bottom to top: A VGG19 architecture, a plain 34 layer CNN and a Resnet34 (He et al., 2015)

The success of Residual Networks has been so significant, that residual learning has become a field of study by itself. Residual connections have also been added to Inception architectures, as in (Szegedy et al., 2016a).

Residual learning

The motivation behind residual network is being able to construct very deep neural network architecture while avoiding the problem of accuracy saturation and degradation. This degradation is not caused by over-fitting, but by difficulties in training networks of such depths. If the solution of a problem is found using a set of layers, adding more layers should not reduce the performance of the model, as the new layers should learn to perform an *identity mapping* of the previous, optimal, layers. This has been empirically shown to be false, as deep networks have difficulties finding such mappings, thus creating the degradation problem (He and Sun, 2015; Srivastava et al., 2015a,b). The solution proposed in (He et al., 2015) is adding a residual mapping, which, in feed-forward networks, are skip or short-cut connections, which can be visualised in the following graph:

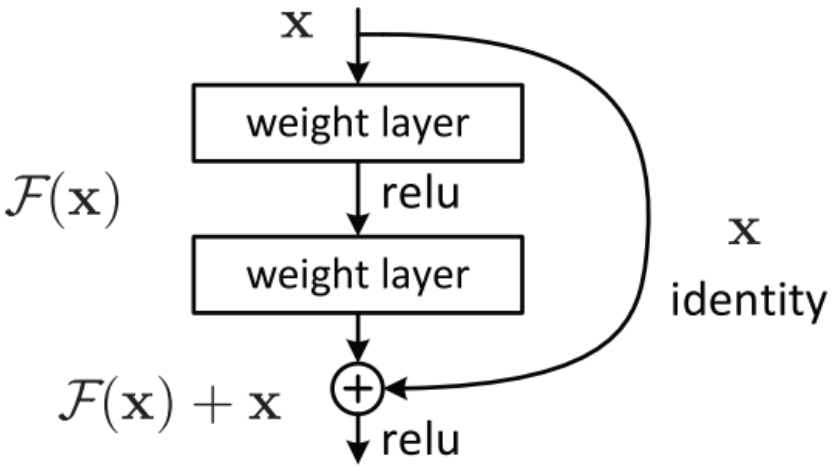


Figure 2.7: Simple residual connection. Obtained from (He et al., 2015)

By adding such residual connections, the network is able to learn identity mappings with ease, which allows the efficient optimization of very deep networks (Dietz, 2017). Residual connections have shown such success that they have been used in top-performer networks, and they are now widely used in the CNN literature (Han et al., 2017; Yunpeng et al., 2017; Kim et al., 2016; Zhang et al., 2017; Yu et al., 2016; Lim et al., 2016; He et al., 2016; Laina et al., 2016; Xie et al., 2017; Zhang et al., 2016)

2.2 Machine learning in photography

The advancements made in the image recognition literature have also made an impact in digital photography. For instance, by applying convolutional neural networks to short-exposure and low-light images, some authors were able to significantly improve traditional low-light picture enhancement techniques (Chen et al., 2018). In (Fang and Meng Zhang, 2017), the authors propose a system capable of enhancing photographs by automatically cropping and post-processing them, thus finding, from raw, unedited images, a better set of composition, illumination, contrast, colour enhancement, etc. They solve this task with the help of Generative Adversarial Networks (to generate the *dramatic mask*) and an Inception V3 discriminator, used to check that the modifications made to the picture actually enhanced its aesthetic value. Another example can be found in (Gharbi et al., 2017), where a the authors propose a system capable of automatically enhancing pictures in real time, using convolutional neural networks. This is already a part of modern smartphones, like Google Pixel [1]. Other smartphone manufacturers like Huawei use machine learning to categorize each picture taken by the phone, and, depending on the category, they perform different enhancements to the picture [2].

Another interesting use of machine learning in photography is *style transfer*. Style transfer is a technique used to transform pictures so that the content of the picture is preserved, but the stylistic appearance of the picture matches the style of another picture, or even the whole work of a painter. The technique used to solve this task is, again, a deep convolutional neural network (Gatys et al., 2016).

The use of deep learning for photography enhancement is not only a theoretical field of study. It is now an important part of many professional and non-professional software, as it can be found in image editors [3], mobile applications [4] [5], websites for image scaling [6] and even devices that automatically pick the best settings for your camera [7].

[1] https://www.forbes.com/sites/evgenytchebotarev/2018/05/30/ai-is-already-changing-the-way-we-think-about-photography/

[2] https://www.theverge.com/2018/4/6/17198708/huawei-p20-pro-review

[3] https://www.adobe.com/sensei.html

[4] https://prisma-ai.com/

[5] https://cloud.google.com/vision/

[6] https://letsenhance.io/

[7] https://witharsenal.com/

2.2.1 Computational aesthetic assessment in photography

This section is based on my IPP report. The popularization of convolutional neural networks have created the possibility of designing and training computational methods capable of understanding aesthetics in photography. Consequently, in the recent years, several studies interested in this topic have been published.

Many times, as in many other machine learning problems, authors make use of a task-specific dataset that allows them to compare models. In aesthetics assessment, those datasets are usually a list of pictures, rated by a set of human perceivers. Additional information about the pictures may include manually annotated data of the contents or styles of the pictures, or information about the users that rated each picture.

One of the most important aspects in aesthetics assessment is the way in which ratings are analysed. Sometimes, authors create classifiers that are able to distinguish between pleasant and unpleasant pictures, which is a classification problem; whereas others prefer to assign each picture with a real-valued number, which is a regression problem. As it was seen in previous sections of this report, classification problems are evaluated differently than regression problems.

It is possible to divide the literature on this topic into two parts, depending on the whether or not individual preferences are taken into account. There are many studies that fall into the second category. In those studies, the goal is typically to predict the mean rating that a picture would receive. For instance, in (Appu Shaji, 2016), the author proposes a model that tries to represent pictures in a latent space in which the distance between aesthetically pleasant pictures is smaller than the distance between an aesthetically pleasant and an unpleasant picture. To do so, they combine a triplet hinge loss with a VGG-13 network, and train their model using 100000 images. In (Lu et al., 2014), a shallow double-column convolutional network is trained to classify images as pleasing or unpleasant. The difference between their model and other authors' models resides in the fact that their model combines a global view of the whole image with a local, cropped, view. According to the authors, this double view provides a higher prediction accuracy. Aesthetics-preserving cropping is the focus of study of (Wang and Shen, 2017), a study that proposes the use of a convolutional neural network to optimally crop a picture so that its aesthetics value is maximized. This topic was also studied in (Chen et al., 2017). There are many other studies that study image aesthetics and related topics using convolutional neural networks, like (Bianco et al., 2016; Deng

et al., 2017; Yu and Chen, 2018; Denzler et al.; Kong et al., 2016).

There are articles that do not make use of convolutional neural network, and instead they use other machine learning algorithms, like SVM or decision trees, along with hand-crafted features that are typically inspired by the psychology or photography literature (rule of thirds, illumination, etc.), to predict the aesthetics value of pictures, like (Datta et al., 2006). More examples of the use of this kind of features can be found in (Niu and Liu, 2012; Jin et al., 2010; Vogel and S. Khan, 2012; Jiang et al., 2010; Yan Ke et al.; Luo and Tang, 2008; Bhattacharya et al., 2010). These studies are understandably older than the studies that use CNNs, and their findings are less generalizable as they require to manually choose what features to look for. It can be argued that the emergence of CNNs have eliminated the need for this manual choice of features.

Datasets One of the most important decisions when studying an empirical machine learning problem is the choice of the dataset to use. This choice defines the limits and the scope of the study. There is a tendency in this field of creating increasingly large datasets, so the results can be more generalizable and the features found by the models can be more meaningful. As in image classification, there are datasets that are widely used in the aesthetics assessment literature, among which some can be highlighted (Deng et al., 2017):

CUHK-PQ This dataset, proposed in (Tang et al., 2013), has around 18000 images, each with a binary aesthetic value and a set of categories assigned to the pictures related to their content.

DPChallenge This dataset contains around 16000 images, and a rating distribution from 1 to 10. It was introduced in (Datta et al., 2008).

Photo.Net This dataset, similar to *DPChallenge*, proposed in (Joshi et al., 2011), has around 20000 pictures, rated from 0 to 7.

AVA The Aesthetic Visual Analysis dataset contains around a quarter of a million pictures, each rated by at least 78 different people, with ratings from 1 to 10. In addition to the ratings, each picture is also categorised in stylistic and content classes. Due to its size and the amount of information available of each picture, it is the most common dataset in the aesthetics literature. It was proposed in (Murray et al.).

Personalised aesthetic assessment in photography

Personalised predictions of the preferences of a user is one of the main concerns of collaborative filtering, which is the basis for many recommender systems. It is based on what similar users have liked in the past, without taking into account the contents of the items themselves. This is limited, as the prediction of the rating of new items cannot be performed until a set of users have given feedback on them. In image aesthetics prediction, this will require to ask a set of users to rate new pictures, so that the rating that other users would give to those pictures could be predictive. Among the techniques used for collaborative filtering, matrix factorization algorithms like Factorization Machines stand out due to their popularity (Schafer et al., 2007; Isinkaye et al., 2015; Hong et al., 2013). There are methods that can combine image contents with matrix factorization or other collaborative filtering methods (O'Donovan et al., 2014; Rothe CVL et al., 2016), but those methods are limited as they still require several ratings of each picture so that they can predict new ratings of those pictures. This requirement can be prohibiting, as in many real-world applications the pictures that one user rates can only be seen by that user.

To overcome those limitations, and to incorporate the potential of convolutional neural networks in personalised image aesthetics predictions, the authors in (Ren et al., 2017) propose a residual-based model that is capable of solving this problem. Their model combines a Convolutional Neural Network (a version of the CNN in (Ioffe and Szegedy, 2015)), which computes a *mean rating* prediction, that does not take into account individual preferences, and a Support Vector Regressor (SVR), which computes an offset prediction (how different the user's ratings will be compared to the mean rating). The SVR does not take into account the input image itself, but it considers data about its contents and style (predicted using different neural networks):

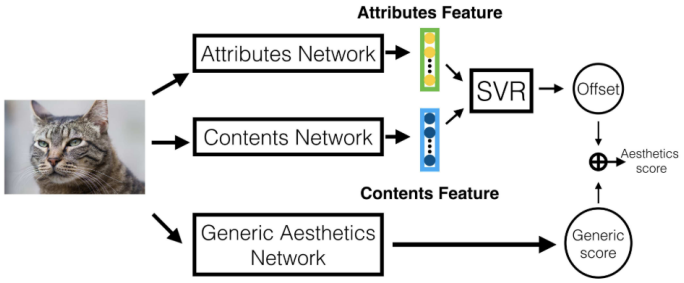


Figure 2.8: Residual model proposed in (Ren et al., 2017)

This model, called *Personalised Aesthetics Model* (PAM) allows the personalised prediction of image aesthetics ratings, and it does not require the ratings of more than one user. This is because the generic aesthetics network in 2.8 is trained off-line, and, once it is trained, it does not require more human input. The model, while powerful, is limited in some ways. For instance, it requires the training of 3 neural networks with different purposes, and a SVR for each particular worker in the dataset. This can be hard to design and optimize, and may be computationally inefficient. Moreover, the use of the SVR makes it harder to automatically optimize all the parts of the model at once (although in their approach, they do not need to do so, it may be beneficial in some cases, as in picture enhancement using gradient ascent). Finally, there is an assumption that all the differences between the aesthetics preferences between users can be understood by preferences over contents and styles, using a fixed set of said attributes. This greatly limits the type of patterns that the model that explain individual preferences over aesthetics in photography will be able to find.

In addition to this proposed model, Ren et al. propose an active learning approach so the model can improve itself when the user gives feedback on more images. Furthermore, the authors also present two datasets:

REAL-CUR This is a small dataset with 14 personal picture albums, whose ratings are provided only by the owners of the albums themselves.

FLICKER-AES This is a bigger dataset, with around 40000 images downloaded from Flickr and rated by 210 Amazon Mechanical Turk workers. The images are rated from 1 to 5 and the mean rating of each picture is the mean of the ratings that the workers gave to that picture. For the test set, 37 workers are selected that have labelled between 105 and 171 ratings, which have labelled about 4373 images in total. Neither those images or those workers are part of the training set, so there is no overlap in the workers or the pictures in the two datasets. This is a much bigger dataset that REAL-CUR or other alternatives like the one presented in (Kong et al., 2016).

Summary Thanks to the dataset FLICKER-AES, which will be described with more detail later in this report, and the findings of (Ren et al., 2017), it is now possible to create machine learning methods capable of predicting personalised image aesthetics ratings. Nevertheless, the findings in (Ren et al., 2017) have limitations. Most notably, they use data of the contents and styles of the pictures and the PAM model they propose

combines a SVR with a CNN. The main hypothesis of this thesis is that it is possible to embed personalised preferences in one convolutional neural network, while increasing the variety of aesthetic-related features that can be found, simplifying the model and possibly improving its overall prediction accuracy.

Chapter 3

Methods

3.1 Data augmentation

Data augmentation (Wong et al., 2016) is a technique widely used by machine learning practitioners which has the objective of increasing the size of the training dataset, so as to improving the generalization capabilities of a machine learning model. Data augmentation involves generating new data examples from the examples that are already present in the training dataset. In image classification, in which the input data are images, the data augmentation methods must preserve the contents of the pictures so that the features that are useful for identifying an object are present in the generated images.

There are two ways of performing data augmentation when training machine learning models. Generating data *off-line* involves generating the fake images before training the model, whereas generating data *on-line* involves generating the new images while training, which is particularly useful in mini-batch training. Most modern machine learning frameworks incorporate on-line data augmentation as a built-in option (Raj, 2018). In computer vision, some methods stand out due to their popularity:

Flipping images This involves reversing an image across one of its axis (horizontal or vertical).

Random rotation As the name suggest, this technique involves rotating the input image a random number of degrees. Rotation can also be performed in a fixed number of degrees (e.g., 90 degrees) but in a random number of times. This technique is only useful when the object that the model needs to identify does not change its label when

it is rotated. For instance, rotating a picture of an 8 can create an image that has the ∞ symbol, which is not the same symbol.

Random cropping This technique selects a random rectangular section of the input image.

Rescaling Rescaling an image is changing its dimensions, usually keeping the totality of their contents. Usually, the objects in the rescaled image do not have the same proportions as in the input image.

Adding random noise This technique adds some random (usually gaussian) noise to each of the pixels in the input picture, thus modifying its appearance.

Advanced techniques Some new methods, such as GAN data augmentation, create completely new images by using a generative adversarial network (Antoniou et al., 2018). They offer the opportunity of generating images with a much broader variety than traditional data augmentation methods. Other authors propose pairing sample images, which reportedly adequate for tasks with a very limited amount of data (Inoue, 2018).

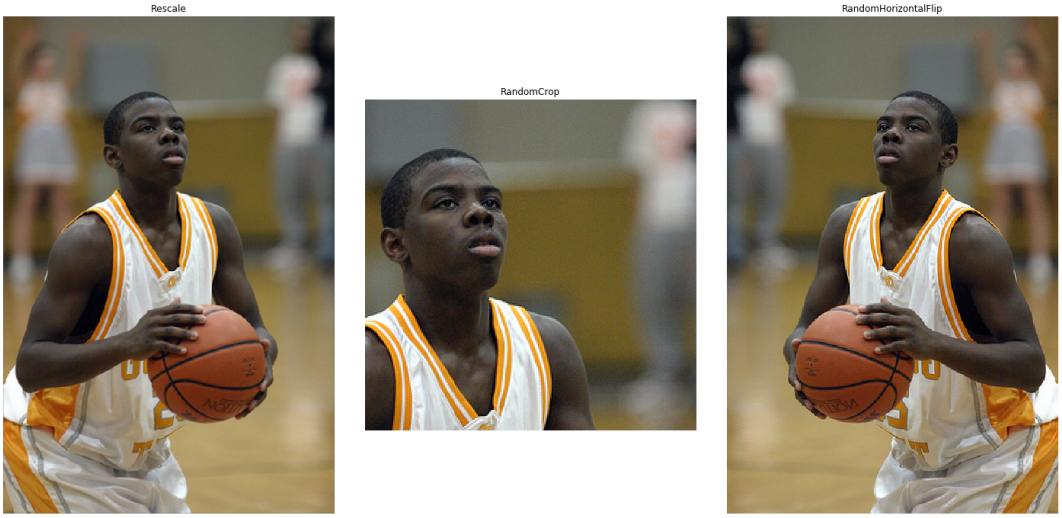


Figure 3.1: Visualization of some data augmentation methods (rescaling on the left, random cropping on the center picture, random horizontal flip on the right). Picture obtained from the FLICKR-AES dataset, with a mean rating of 3.6/5

The choice of what data augmentation method to use is largely dependent on the task that the model needs to solve. Consequently, for image aesthetics prediction, there are methods that are not adequate, as they may transform the aesthetic value of the pictures. It is worth noting that, even when the input is modified in some way by data augmentation methods, the target value stays the same, so it is important to be careful when using those methods. There are also other factors that need to be taken into account. Some methods may be more effective than others, but also more time-consuming (Wang and Perez, 2017).

Many of the methods listed above can be concatenated. For example, given an input image, it is possible to sequentially rescale it, crop it, flip it and add random noise to it. This can be performed in any order. The more methods that are used, the greater the variety of pictures the network will be trained on, which has a positive impact on generalization.

Figure 3.2: Image augmented with conditional generative adversarial networks. The main object of the pictures is recognizable, but their appearance is different. Obtained from (Wang and Perez, 2017)

For image aesthetics, some methods are not adequate. Adding random noise to a picture can completely change its appearance, meaning that the transformed image, if shown to a set of human perceivers, they will rate the picture differently than the original picture. Methods like GANs may not appropriate either, as it may be necessary to have human supervision on the generated images (however, this is beyond the scope of this thesis and will not be empirically tested). Vertically flipping or rotating a picture can also distort its appearance, which is the most important factor in image aesthetics. Therefore, in the rest of this thesis, the only data augmentation methods that will be used are horizontal flipping, rescaling and random cropping. It can be argued that random cropping and rescaling can have a significant impact on the appearance of the pictures, but the results when using those methods were better than without using them, which is the reason why they will be used in the experiments of this thesis.

3.2 Transfer learning

Transfer learning (Torrey and Shavlik, 2009) is a popular machine learning method where the knowledge acquired to solve one problem is re-used to solve another, usually similar, problem. The main goal is to use the information and features obtained when solving the first task to improve the generalization capabilities of the model that is used to solve the second task. The more similar the two tasks are, the more appropriate transfer learning will be. As data augmentation, this method is used to solve the problems that small datasets create for deep learning models.

In deep learning and, in particular, in convolutional neural networks for image recognition, transfer learning can be done in two different ways (Browniee, 2017). The first way involves selecting the original task, train a CNN model to solve that task, and use the trained model as the starting point for the second task. The second way is selecting an already trained network (e.g., a Resnet or VGG) and tune its parameters so it can solve the second task. This second approach can be thought of as a way of initialising the weights of the network so their configuration is closer to the solution of the second task than a random initialization would be. This is because many of the features that were learned to solve one computer vision problem can be reused to solve another, slightly different, computer vision problem. For instance, early layers of CNNs usually learn filters that look for edges or corners in the input images. This is generally the case for many computer vision problems, and it is not necessary to re-learn those filters for each task. Another reason why using pre-trained networks is a good idea is the fact that those networks are trained for weeks using powerful computers, and tuned so that they obtain a state-of-the-art performance. This kind of computational resources may not always be available.

The way to tune the parameters of the pre-trained model depends on the size of the dataset of the second task, and how similar the two tasks are. A general approach to perform transfer learning in CNNs is to use the convolutional layers as fixed feature extractors and only re-train the linear classifiers at the end of the CNNs. This approach finds a different combination of features that is best to solve the second task, while re-using the features learned in the first task. This is typically the case in problems where the dataset of the second task is small but the first and second tasks are similar. When the tasks are similar but the second dataset is large enough, it is also a possibility to tune all the parameters of the network. When fine-tuning a pre-trained model, using smaller learning rates is advised. (Karpathy, 2018).

In many deep learning frameworks incorporate compatibility for the use of pre-trained machine learning models, trained for many different tasks (Jia, 2018). For fine-tuning in image recognition tasks, the typical approach is to use a deep CNN trained on the ImageNet dataset, which contains more than 14 million annotated pictures. Training deep CNNs with a dataset of that size with standard computational resources is infeasible, but the developers of VGG, Inception or Resnet have shared their trained models for any deep learning practitioner to use for free.

One of the main assumptions of the work of this thesis is that the features that networks like Resnet learned when trained in the Imagenet dataset can be used as a starting point for image aesthetic prediction tasks. Consequently, it is possible to tune a pre-trained Resnet so it can learn to adjust image classification features into a aesthetic-related features. This is a common practice in the image aesthetic prediction literature, as it is suggested in (Ren et al., 2017; Deng et al., 2017; Jin et al., 2016; Murray and Gordo). The way transfer learning is used in this thesis will be explained with more detail in the experiments section of this thesis.

3.3 Multi-domain learning and Multi-task learning

Multi-domain learning (MDL) refers to the problem of using a single model that can solve the same task in different domains (Nam and Han, 2016). This is related to Multi-task learning (MTL), which is the problem of using a single model that can learn different tasks at the same time within the same domain. In MDL, one approach is to have a single neural network that can represent all of the domains, which also includes a set of domain-specific *adapters* that adjust the network to each domain. This is what was proposed in (Rebuffi et al., 2017, 2018). By using *residual adapters*, they are able to train networks that can work in different domains, with a significant amount of inter-domain parameter sharing, with the addition of small residual adapter, domain-specific, modules that adjust the filters in the convolutional layers for each domain, without significantly increasing the size of the network. They found that the optimal configuration of such adapters is in parallel to each convolutional layer of the Resnet. They provide additional information about the best regularization methods to train those adapters efficiently, and a empirical justification on where to put those adapters.

Residual adapters as a model of user-specific aesthetic preferences In this thesis, we argue that those adapters can be used to adjust the features learned by a user-agnostic image aesthetics prediction network to the preferences of each user. The user-agnostic network would work as a mean aesthetic predictor, and the adapters of the network, specific for each user, will be used to model the preferences of each user within the network. This method, compared to standard finetuning of the fully connected layers, or compared to the method in (Ren et al., 2017), allows to find individual preferences with different levels of abstraction, as all the filters in the model, whose levels of representational abstraction depend, among other factors, in their position in the network (the deeper, the more abstract), will be accompanied by a residual adapter. The advantage of this method is that, by not assuming a fixed level of abstraction in which personal preferences are encoded, it is possible to find more meaningful features that represent each individual's preferences over aesthetics in photography.

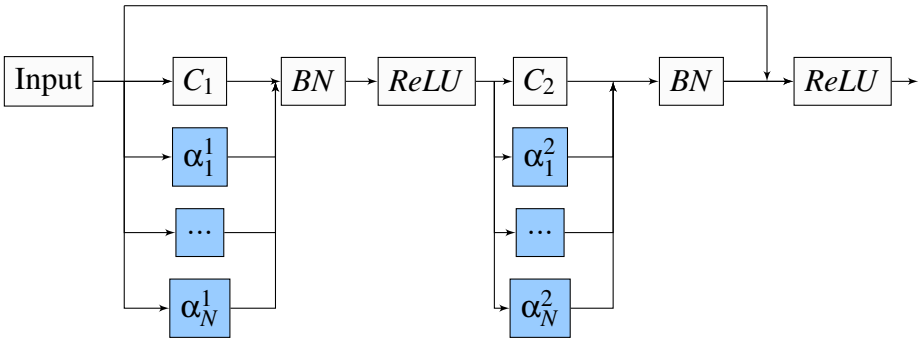


Figure 3.3: *This figure and caption are based on my IPP report and from (Rebuffi et al., 2018)* Illustration of the proposed parallel personalized residual adapters and their position in a typical Residual Network module (Zhang et al., 2016). Each of the α_i blocks is a set of 1×1 convolutional filters that is uniquely trained for each user i, of the total of N users. The C_n layers are the same for each user and have the purpose of finding those aesthetic features that are shared by every user. This architecture allows some degree of shared knowledge between the user ratings, while each user remains to be considered individually. Even if α was only illustrated in parallel to the convolutional layers in a typical Resnet module, it is possible to have one user-specific set of adapters α^n in parallel with each shared convolutional layer C_n of all the resnet modules in the network. It is worth noting that, when the adapters' weights have a value of 0, the network with the adapters is strictly the same network as the network without the adapters. Consequently, if a proper regularization technique is used (e.g., weight decay, which could make the weights of the adapters get close to 0), the network with the adapters should show at least the same performance as the network without them.

3.3.1 1×1 convolutions

There are several reasons authors in the machine learning literature have used 1×1 convolutional filter sizes, most notably dimensionality reduction. They are an important part of Inception architectures, as they are used to keep a reduced representation within each inception module, thus avoiding computational bottlenecks that can damage performance in very deep Inception architectures (Szegedy et al., 2014).

Nevertheless, there is a more important reason behind using 1×1 convolutional filter sizes in our proposed personalised aesthetics prediction method, which is to keep a reduced number of user-specific parameters in the network. The features that are used to predict aesthetics in the network should be learned by the user-agnostic layers, and the adapters should learn to predict how important each feature is for each user. This reduces training time, and can avoid overfitting when the number of images rated by each user is small. As was argued before in this document, the goal of the adapters is to change how important each feature is for each individual user, thus finding combinations of feature maps with different levels of abstraction.

3.4 Evaluation metrics

As the main goal to be solved by the proposed models is to predict the aesthetics ratings of pictures, using a number within a given range (e.g. 1-5), there are two approaches that could be taken. First, it is possible to construct a classifier which would return as an output the most likely rating (1,2,3,4 or 5). This approach is not suitable when computing a mean rating, which can be real-valued, so the networks should learn to perform regression on the aesthetic value of pictures. There are many ways to evaluate regression methods, but the literature on this topic typically uses two: Mean Squared Error (squared difference between target value and predicted value), which was explained with more detail in the loss functions section in this document; and the Spearmans's ρ, which is used in (Kong et al., 2016; Ren et al., 2017).

The following paragraph is taken from my IPP report. This evaluation metric will quantify the correlation between the predicted subjective picture ratings and the ground truth ratings found in the test dataset. This metric has the advantage of being bounded in the $[-1, 1]$ range, it is easy to interpret (higher values of $|\rho|$ indicate a higher correlation) and it is invariant to monotonic transformations of the ratings, which means that a modification in the scale of the ratings (e.g., from 1-5 to 0-10) will have no impact on

the value of ρ. Let N be the number of images ranked, r_i the real ranking of the picture i when ordered by the ground truth list of scores $\{y_i\}$ and $\hat{r}_i$ the estimated ranking of the picture i when ordered by the estimated list of scores $\{\hat{y}_i\}$, then:

$$\rho = 1 - \frac{6 \times \sum_{i=1}^{N} (r_i - \hat{r}_i)^2}{N^2(N-1)} \tag{3.1}$$

Spearman's ρ measures the monotonic relationship between two variables. This is different to other evaluation metrics such as Pearson's correlation, which assumes a linear relationship between the variables (Minitab, 2017).

Chapter 4

Implementation

4.1 Dataset description

The dataset used throughout the experiments in this thesis is the FLICKR-AES dataset, presented in (Ren et al., 2017), which can be downloaded at `https://drive.google.com/drive/folders/1XLlPu_lgHqRstF7DBmXQ2QPSK9KPx1Yu`. It contains 40500 images rated by 210 *workers*, with ratings in the 1-5 range. Each tuple (image, user, rating) contains integer ratings, but the authors also provide the mean rating for each file. The distribution of mean ratings, when normalised into the 0-1 range, is:

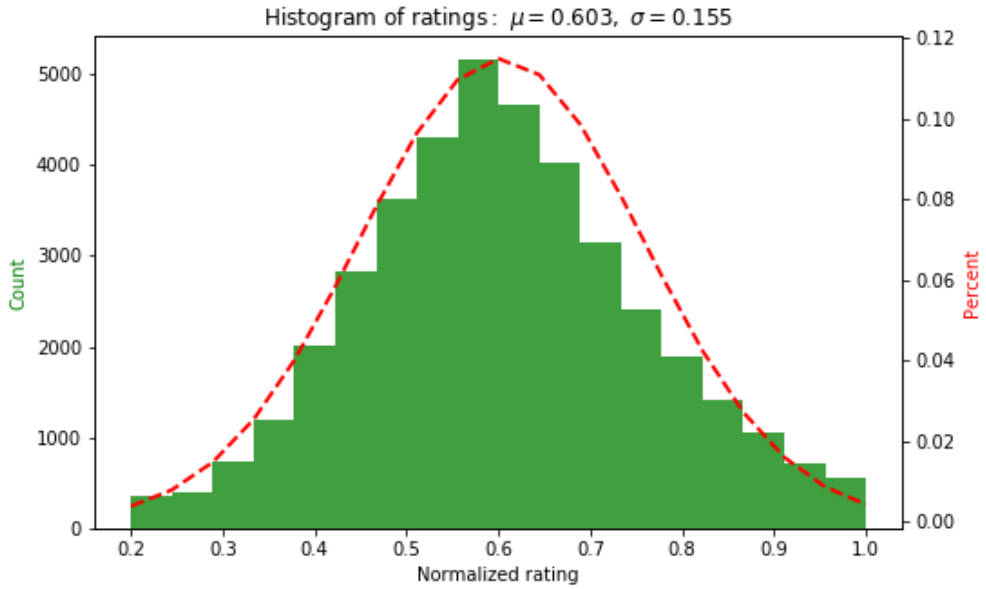


Figure 4.1: Histogram of the FLICKR-AES dataset. It approximately follows $p < 0.001$ a normal distribution with mean=0.603 and standard deviation 0.155

Each picture is rated, in average, by 4.89 workers, with an standard deviation of 1.87. The maximum number of times a picture was rated was 48 times, and the minimum was 1. There is not a significant correlation between the number of times each picture was rated and its mean rating (correlation coefficient of 0.03). The distribution can be seen in the following plot:

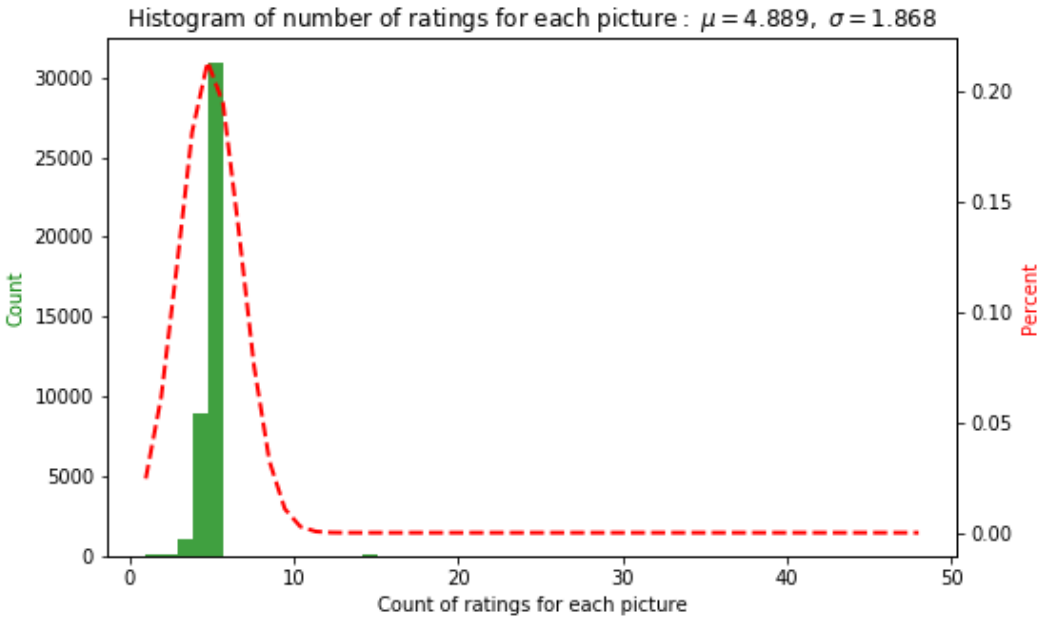


Figure 4.2: Histogram of the number of times each picture was rated in the FLICKR-AES dataset It approximately follows $p < 0.001$ a normal distribution with mean=4.89 and standard deviation 1.87

There is a wide variety of themes (portraits, landscapes, etc.) , styles, contents and topics in the pictures of this dataset. The dataset includes pictures in grayscale, with different kinds of blurs, lighting conditions, etc. This variety should allow the algorithm to learn features that are related to a wide variety of aesthetics characteristics, regardless on the contents, styles or themes of the pictures themselves. Consequently, said variety will increase the robustness of the network when it sees images for the first time, compared to other datasets with a smaller variety of pictures.

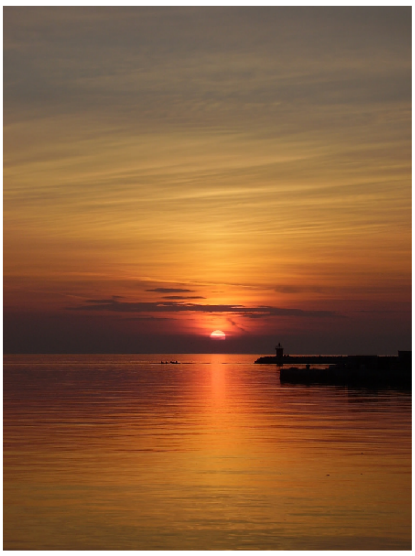


Figure 4.3: Picture with the highest mean rating of the FLICKR-AES dataset, with a 5/5

Figure 4.4: Picture with the lowest mean rating of the FLICKR-AES dataset, with a 1/5 (minimum rating was a 1)

4.1.1 Train/Test division

To make the results of the experiments comparable to (Ren et al., 2017)'s results, the chosen train/test selection was performed in a similar way. It was not possible to find the exact same train/test split, but we are confident that the division is almost identical. 37 workers were selected to conform the test set, they had rated 4977 pictures, with a range of number of images rated per worker of 112 to 180, with a mean of 138.89. This is slightly different to what was reported in (Ren et al., 2017) (range of 105 to 171, mean of 137, and a total of 4737 images for the 37 workers). The train dataset has the remaining 173 workers, and there are no pictures in the train set that are also in the test set. This way, there is no overlap between the two datasets, making the evaluation of the models more fair, as the test set will be representative of both unseen workers and unseen pictures. Ratings were re-normalised for the experiments before performing the split, into a 0-mean unit-variance configuration. The difference between the distribution of ratings in both datasets can be seen in the following plot.

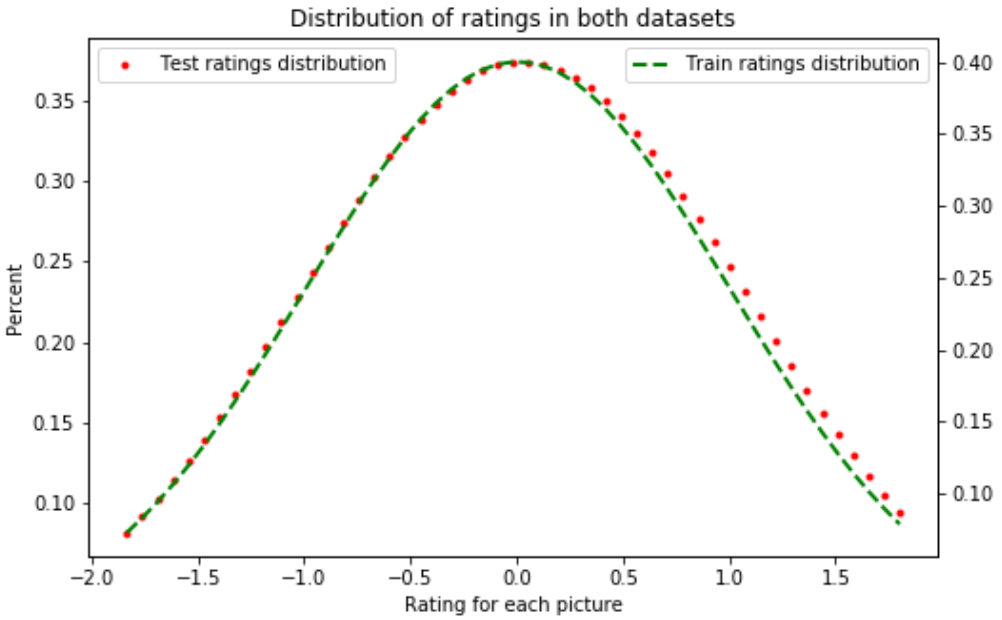


Figure 4.5: Distribution of the ratings in the train and test datasets. It is visible that the division has not introduced a significant difference between the two datasets. The test dataset follows a normal distribution of $\mu = 0.027$, $\sigma = 1.06$ and the train dataset follows another normal distribution of $\mu = 0.00489$, $\sigma = 0.996$. With a two-sided Kolmogorov-Smirnoff significance test (Panchenko, 2006), we obtain a $p < 0.0001$. We can assume that two ratings distributions are drawn from the same continuous distribution. This is important, as the train/test division has not introduced any bias with respect to the mean rating of each picture, and the variance of both distributions are also very similar. It this was not the case, then the behaviour of the workers in both datasets would be significantly different, and the possibility of inferring the preferences of the workers in the test set using the learned preferences from the workers in the train set would be questionable.

The pictures in the dataset were stored using the *JPG* format, and the two associated tables stored in *.txt* format, which were read as comma-separated values.

4.2 Deep learning frameworks

There are several deep learning frameworks that are useful for designing, training and testing deep neural networks. Among them, the most popular are Tensorflow[1], Keras[2] and PyTorch[3]. There are other popular frameworks, like Caffe [4] or Mxnet [5]. The choice between Tensorflow, Keras or PyTorch involves taking into account many factors, like performance, flexibility, code re-usability, ease of use, etc. Keras is a framework designed to provide a high-level API for deep learning research and application. It is now the official high-level API for Tensorflow[6]. The use of parallel adapters can be tricky in Keras, as it is not incorporated as a built-in function and creating custom layers and computational graphs in Keras can be challenging. Other experiments made in this thesis could have been made in Keras, but to facilitate code reuse, all of the experiments were performed using the same framework.

The decision between PyTorch and Tensorflow, once Keras was discarded, was made taking into account three factors. First, Pytorch allows the creation of dynamic computational graphs, which means that it is possible to change the computational flow of tensors while training. This is not possible in Tensorflow and it could have created burdens when training the residual adapters. Second, the built-in methods in PyTorch for training, loss and gradient computation, data augmentation, mini-batch creation, and other deep learning-related factors seemed more intuitive than Tensorflow's, which was an important determinant in favour of PyTorch. Finally, the authors of (Rebuffi et al., 2017, 2018) provided example code in Torch for the residual adapters, which inspired the code used to train the models in this thesis (although they were coded from scratch). Nevertheless, Tensorflow is generally better documented and it is considered to be better for deploying methods. Deeper comparisons between those frameworks can be found on (Hannun, 2017; Dubovikov, 2017). Performance was not taken into account as the GPU and Cuda version (most important factors in performance when training deep models) used both in Tensorflow and PyTorch would have been the same.

[1] https://www.tensorflow.org/
[2] https://keras.io/
[3] https://pytorch.org/
[4] http://caffe.berkeleyvision.org/
[5] https://mxnet.apache.org/
[6] https://www.tensorflow.org/guide/keras

4.3 Data processing

All models were trained using mini batches, with diverse sizes depending on the task, model, etc. To do so, the following process was followed. First, a custom dataset class was created, which returned an image and its corresponding rating (and the user that gave that rating, for the experiments that took users into account). Images are read from a local directory using the Python Imaging Library and its API Pillow[7]. The images were read as a three-channel JPG image and translated into a 3-dimensional Numpy [8] matrix. This process was done so that gray-scale images, usually stored as a 2-dimensional matrix of intensity of pixels, could have the same shape as colour images. Numpy was chosen as a temporary way to store the pixels information due to its efficiency and capabilities of performing operations on matrices.

Five custom image transformation methods were created. PyTorch built-in data augmentation methods were not designed for custom regression tasks, but the code was easily adapted from their built-in methods:

Normalize Pictures' pixels were normalized using Imagenet's standard deviations and mean values for each of the three RGB channels. This is a common practice in image classification research, as the normalization values for different problems are usually similar, and computing the mean and standard deviation of each channel in a new dataset can be expensive. Pre-trained models expect normalised inputs, and the way PIL reads images (intensities from 0 to 255, for example) is not suitable.

Rescale This method receives an image of a given dimension and returns the same image with another, fixed, dimension. This method was created to force all of the images to have the same dimensions, as pre-trained models expect images with a given dimension, usually 224×224 pixels. This method changes the proportions of the objects in the images, but it is necessary for the use of pre-trained models.

Random Horizontal Flip As the name suggests, this method receives an input image and, with a given probability p, it returns the same image flipped horizontally. This method was created as a data augmentation policy that preserves the aesthetic value of the images, as it was argued earlier in this thesis.

[7] https://pillow.readthedocs.io/en/latest/
[8] http://www.numpy.org/

Random crop This method receives an input image and returns a randomly chosen subsection of the image, with a given dimension. This method was also used as a data augmentation technique that, even if it could be argued that it does not preserve the aesthetic value of the whole picture, it showed to work well in practice and was also used by (Ren et al., 2017).

To tensor transformation This method receives an image in a Numpy format and it returns it as a tensor for the use of PyTorch models. The previous operations on the images were performed using Numpy due to its efficiency but PyTorch models expect tensors as their input. The rating of the image is also transformed into a one-dimensional tensor.

Then, two *data-loaders* were defined, which are the PyTorch method of creating mini-batches during training and feeding them to the networks. The training data-loader performed the following operations on the pictures: First, it re-scaled the input picture to a 256×256 picture, then it randomly flipped it with a probability of 0.5, then randomly cropped it into 224×224 pixels, normalised it and finally transformed it into a tensor. This process of rescaling to a 256 pixels square matrix, then cropping it into a 224 pixels square matrix is a common practice in image classification, and was also used by (Ren et al., 2017). Normalising the image after randomly cropping it was done because the cropped image is smaller than before cropping it, and normalising the image is computationally more expensive if the image is larger. The random horizontal flip could be done at any time in the data processing flow before the tensor transformation.

For the test data-loader, no data augmentation is necessary, so the only transformations performed on the images were rescaling them into 224×224, then normalise them and finally transforming them into a tensor. To create the minibatches, the datasets were shuffled randomly and a modification was made so that, in case the current image could not be read from the directory for any reason, the data loader did not include an empty image in the mini-batch. The size of the minibatch was chosen differently for each experiment, always with the goal of maximising GPU memory usage during training, but also taking into account other factors, such as the number of images available per worker, etc.

Chapter 5

Experiments

5.1 Introduction

In this section, the experiments performed throughout this dissertation will be outlined. The results obtained in those experiments will be shown in another section of this thesis, but some experiments will be built on the results of previous experiments, so they are not independent of each other.

The main goal is to find a neural network architecture that can predict personalised aesthetic ratings of pictures, without the need for data about the contents or styles of the pictures or a personalised SVR on top of neural networks. Our model should be able to find personalized aesthetic-related patterns that do not necessarily depend on the contents or styles of the pictures themselves. The process of building such a network will be as follows. First, a baseline architecture will be trained, which will have the objective of finding the mean rating that a set of users would give to a given picture. Then, building on that network, two sets of experiments will be performed.

The first set will assume that new, unseen users, have rated $k = 10$ unseen pictures. Experiments will be carried out to check if the baseline network can be used in addition to the information of those 10 ratings so we can personalize the aesthetic predictions. This experiment is useful for real-world problems in which the number of available ratings per user is limited. We argue that residual adapters are not adequate for problems with this limited amount of examples, as 10 images are not enough to adjust features with different levels of abstraction.

The second set of experiments will assume that we have 100 image ratings available for each user. This setting is more adequate for situations in which the amount of pictures rated by each user is larger, as in image recommender systems, user albums or social networks.

For all the experiments in this section, the data processing method that was used was the method described in the *implementation* section of this document. As it was described there, using random cropping always lead to a better performance of the models. This was the case for all the experiments. Image ratings were normalized into a 0 mean, unit variance configuration, which is a common practice in machine learning. As the results of the models will be evaluated using the Spearman's rank correlation, which is invariant to monotonic transformations like this kind of normalization, this process does not have any impact other than making the models work better.

The learning rule used in all of the experiments was Adam, initialised with the default learning rate of 0.001 and maintaining the recommended momentum and β_1, β_2. As it was argued earlier in this report, Adam is quite robust to the initialization of the learning rate, and this was shown to be the case for the experiments performed in this dissertation. Nevertheless, the learning rate was exponentially decayed by a factor of 90% every 2 epochs, so that learning could be stabilised during training times. The criterion that the network will learn to minimize (the loss function) is the Mean Square Error. Due to the limited amount of data available, the experiments will make a heavy use of transfer learning methods.

5.2 Mean aesthetic prediction network

To create the mean aesthetic prediction network, the train dataset described in the implementation section was used. That is, the network was trained using all the ratings that the 171 workers gave to around 35000 pictures, without taking into account the identity of the workers, with the goal of obtaining a model capable of partially understanding the preferences of the 37 users in the test set. To do so, some architectures were tested, and the chosen configuration was a pre-trained Resnet-18 with some added classification layers. The reason behind this was that a deeper network (e.g. a Resnet-34) lead to over-fitting and they were harder to train (they use more memory, thus the mini batches had to be smaller, increasing the instability of the gradient computation and training times), and shallower networks showed under-fitting. A Resnet-18 was a middle ground that was able to learn aesthetic-related features that could be generalised into unseen pictures and users.

To overcome the limited amount of training data, in addition to the data augmentation methods explained before, the Resnet-18 was pre-trained on the ImageNet dataset. This empirically showed better performance than randomly initialising the weights of

the network, which shows that some of the features that can be used for image classification tasks can be translated into image aesthetic prediction tasks. A reason that can explain this phenomenon is that the image primitives learned in the early layers of the networks, such as edge detection, are largely generalizable.

5.2.1 Network design

The network that showed better results for this set of experiments was a Resnet-18 with a set of added layers in the end. The main component of a Resnet network can be seen in figure 2.7. There are skip connections that bypass the input of an early layer into the input of a deeper layer, usually a layer with 2 levels of depth more compared to the earlier layer. Residual networks make a heavy use of batch normalization, as it is performed after each convolutional layer in the network, as it can be seen in figure 3.3. A Resnet-18 architecture can be visualized in the following figure:

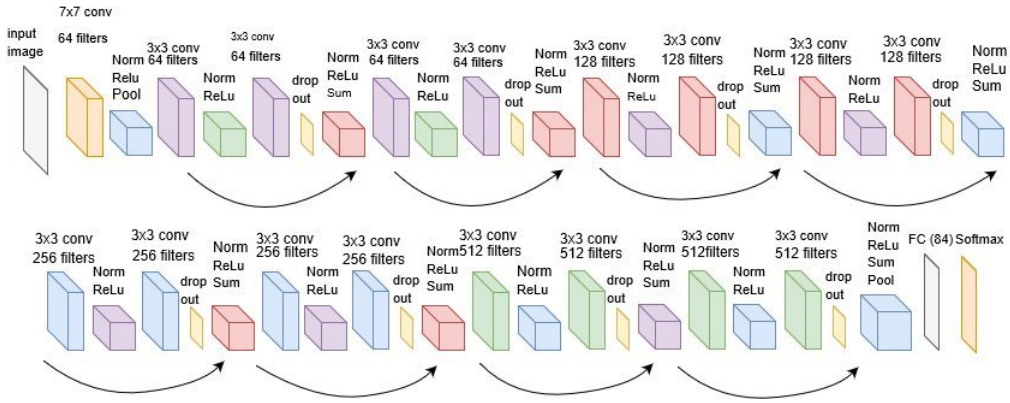


Figure 5.1: Illustration of a modified Resnet-18 architecture, obtained from (Alif et al., 2017). The authors propose the use of dropout after every second convolutional layer. This is not the approach we followed, as the pre-trained Resnet was not trained using dropout there. It can be seen that the first layer is a layer with a larger filter size (7×7), and that the number of filters increases as we go deeper in the network. After each convolutional block, a batch normalization and a ReLU activation are followed. Finally, there is a average pooling layer, with a kernel size of 7. The fully connected and softmax layers at the end of the network are not used in our model, and instead another block of fully connected and non-linear activation layers are trained from scratch. This is a common practice in transfer learning, as the most difficult to train features are those in the convolutional layers, and the last, fully connected layers are easier to train as they simply make use of the latent space learned by the convolutional part of the network.

The full network can be visualized as the following flow of layers:

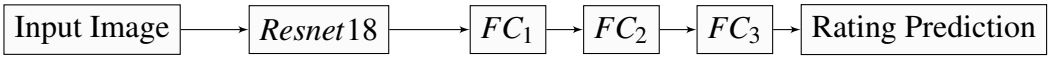


Figure 5.2: Visualization of the baseline network. The network receives an input image, that goes through a Resnet-18, which outputs a vector of 512 values, which transformed by the FC_1 block, which performs the following operations: First, it transforms the 512 values into 1000 using a fully-connected layer (each of those 1000 values is a different linear combination of the previous 512). Those 1000 values go through a PReLU ($a = 0.25$) non-linear activation, a Dropout (p=0.5) layer that is only activated during training, and finally, a batch normalization operation is performed. The following FC_2 block performs a similar operation: It receives 1000 values, and they are transformed into 1000 different values using another fully-connected layer, followed by PReLU, Dropout and batch normalization. Finally, the FC_3 layer receives 1000 values and outputs 1, which is the final rating prediction.

Several factors were taken into account to create this final block of layers. Dropout was used as the only regularization method when training this network. The depth of the block was chosen as 3 fully-connected layers because it worked better than 2 or 1, and adding more layers did not improve the generalization capabilities of the model. PReLU was chosen as the non-linear activation function as it showed slightly better results than baseline ReLU or Leaky ReLU. The number of hidden units in each layer was chosen as 1000 because it showed better results than using less units, and using more units would lead to an increase in the networks parameters, which is something that we wanted to avoid in the personalised prediction experiments.

Description of the experiment

Several experiments were made, with different learning rates, dropout rates, activation functions, etc. We saw that, with the default learning rate of 0.001 and the configuration specified in the previous section, MSE error could be minimized in less than 10 epochs. As the train/test split was fixed, no cross-validation was performed.

5.3 Personalized aesthetic prediction networks

5.3.1 10 images for every user

The goal of this experiment is to understand how much can the network proposed in the previous section learn from a really limited amount of new ratings of unseen users. To do so, for each worker in the test set, 10 pictures will be chosen at random and the last 3 fully connected blocks, which can be seen in 5.2, are fine-tuned. All the tunable layers in those blocks will be tuned, including the batch normalization layers. The fine-tuned model will be evaluated on the remaining pictures of the test dataset that were rated by said worker. This process will be done in a 10-fold validation fashion, so that we can have a more fair estimate of the increase of performance in this model, regardless of the choice of 10 pictures. As in the baseline model, the best model was found before 20 epochs, so the models were trained only for 20 epochs. Those experiments (37 users, 10 networks for each user) were computationally heavy.

5.3.2 100 images for every user

The goal of this set of experiment is to compare methods that can learn subjective preferences of each worker, when there are 100 ratings available of each user. Those experiments will simulate those real-world situations in which there are more ratings available for each user, like in social networks or photography blogs. Three main experiments will be performed. First, as in the case where only 10 ratings were available for each user, only the last block of layers will be fine-tuned. Second, all the network will be fine-tuned and, finally, the adapters will be added to the baseline model so as to check their effectiveness in this domain. The network will be trained with the same settings as stated before, with the exception of the network with the adapters, which will have its own settings. As those experiments are considerably more computationally expensive than in the previous setting as there are 10 times more examples to train the model with for each worker, the network will be trained 3 times instead of 10, thus using a 3-fold cross validation configuration.

Fine-tuning the last layers

The first experiment in this section will be to fine-tune the last block of layers, as in the experiments where only 10 images were available for each worker. All the layers

except for the 3 last fully connected and their corresponding batch normalization layers will be frozen, and trained for 20 epochs.

Fine-tuning all the layers

The second experiment in this section will involve fine-tuning every tunable layer in the network, including all the convolutional, fully connected and batch normalization layers. This method is expected to work better than fine-tuning only the last layers of the network, as more parameters will be modified, but it can also lead to over-fitting.

Network with residual adapters

The last set of experiments in this section will involve the use of residual adapters, as were described in 3.3. We hypothesise that the visual primitives learned by the mean aesthetics model are largely generalizable to new, unseen, users, but the way those filters account for aesthetics may depend on the user. That is, by adding parallel adapters next to each convolutional filter, it is possible to adjust each filter to take into account individual preferences over visual features, regardless on their level of abstraction. This method is different than fine-tuning every layer, as the visual primitives are left unchanged, and only a small set of parameters is added to the network.

This method creates the possibility of having only a baseline model, and a smaller set of adapters for each new user, which is more convenient than having a different fine-tuned network for each user in the system, as it requires much less memory (adding the adapters increases the network size by $10-20\%$ for each user).

In this set of experiments, every 3x3 convolutional layer in the baseline model will have a different set of corresponding parallel adapters, one for each user. As it was easier to implement, one network was created for each user, with the addition of the adapters, but it is possible to have one network with user-dependent adapters that are activated during training or testing. The adapters were initialised with the default Pytorch initialization. In addition to the adapters, the last fully-connected layers of the network were fine-tuned (they constitute a small fraction of the parameters of the network, so they advantage of parameter reuse is preserved), as well as all the batch-normalization layers in the network. Two different optimization methods were used during training the networks with adapters: Adam without weight decay was used to fine-tune the last layers, and, to train the adapters, Adam was used with a weight decay. Several weight decay values were tried, the one that showed better results was 0.05.

In (Rebuffi et al., 2018), it was argued that the best position for the adapters was in parallel to every 3x3 convolutional layer, as it showed the greatest domain-adaptation performance. They argued that the second best option was to put the adapters in the deeper part of the network (last blocks of convolutional layers). In image aesthetics, it could be argued that, as the earlier visual primitives in the network are used as the input for more abstract primitives, those earlier primitives (usually related to edge or corner detection) have less of an impact on the prediction of aesthetics than the primitives found in deeper layers, which are more abstract. To test this, we also designed experiments where adapters were added only to the last 2 or 3 blocks of convolutional layers in the baseline model. Those blocks are the layers with 256 and 512 filters per layer for the first experiment, and 128, 256 and 512 for the second experiment. Every experiment was done in a 3-fold validation fashion with k=100 images per user.

In terms of size, we saw that the network with adapters in parallel to every convolutional layer was $\approx 15\%$ heavier than the network without them. As that size would increase linearly with the number of users for which adapters are needed, it would be necessary, for example, if we had 100 users, to have 100 different networks in the full-network fine-tuning setting, or a single network with a size of 15 times the original network. This parameter re-use can become necessary in systems with a significant amount of users, even more if deeper networks than the resnet-18 were used as a baseline mean aesthetics prediction model.

Chapter 6

Results

6.1 Summary

It was found that our baseline mean aesthetics prediction model obtains a correlation of 0.491 when predicting all the pictures in the test set as a whole. This correlation is increased to 0.531 when considering each worker separately.

By fine-tuning the last layers in the network with k=10 pictures for each worker, an average correlation of 0.575 was obtained. This improvement is greater than what was reported in the active learning method in (Ren et al., 2017).

In situations in which there are 100 training images available for each user, it was found that the best model in terms of average and median correlation was the network with adapters in parallel to each convolutional layer, which obtained a correlation of 0.639. When fine-tuning all layers without adding adapters, a correlation of 0.632 was obtained. The results are not significantly different, but we argue that the use of adapters is still preferable, as it exploits parameter-reuse, thus saving computational resources. It was also observed that the variance of the performance for each worker is greater in the models with the adapters than in the fine-tuning models. We confirm the results in (Rebuffi et al., 2018), where they argue that the best position for the adapters are in parallel to each 3x3 convolutional layer in the network.

Those results on unseen data are better than what was reported in (Ren et al., 2017), where they report an average correlation of ≈ 0.59 on all the pictures in the FLICKR-AES dataset.

6.2 Mean aesthetic prediction network

In this section, the results of the baseline model, which does not take into account individual preferences, are presented. MSE was used to measure the difference between the predicted ratings and the real ratings of pictures. A visualization of the training progress shows that learning in the training set continued after epoch 4, but the error in the test set did not decrease after said epoch.

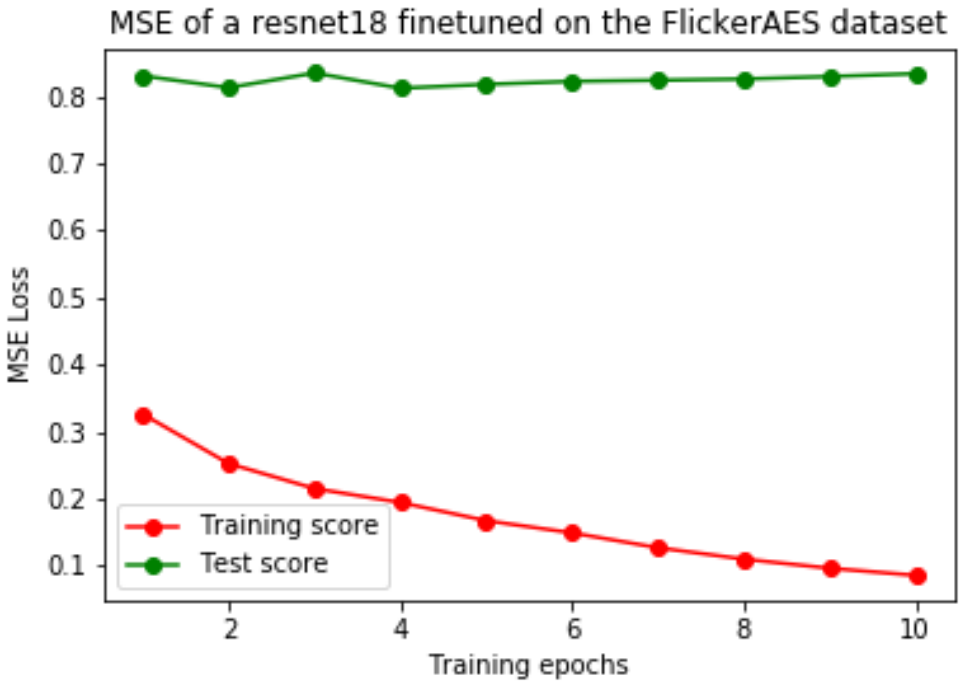


Figure 6.1: Training progress of the baseline model in the configuration that showed less generalization error. The training score shows the mean squared error over all the ratings of all the pictures in the training set, and the test error shows the error over unseen pictures of unknown users. This network can predict the ratings of unseen pictures and users within a certain level of error, but with it shows a significant amount of over-fitting. This is understandable, as the workers and pictures in both datasets are not overlapping, and, as it was argued in this dissertation, preferences over pictures are considerably subjective. Preferences of the users in the training set only explain a percentage of the preferences in the test set. The best validation error obtained was of 0.821.

The network that was chosen as the baseline model from which to construct further models was the network that showed the best results, at epoch 4.

The difference between predicted rating and actual rating can be seen in the following plot. The Spearman's ρ is of 0.491, and the Pearson correlation is of 0.504, which is slightly greater:

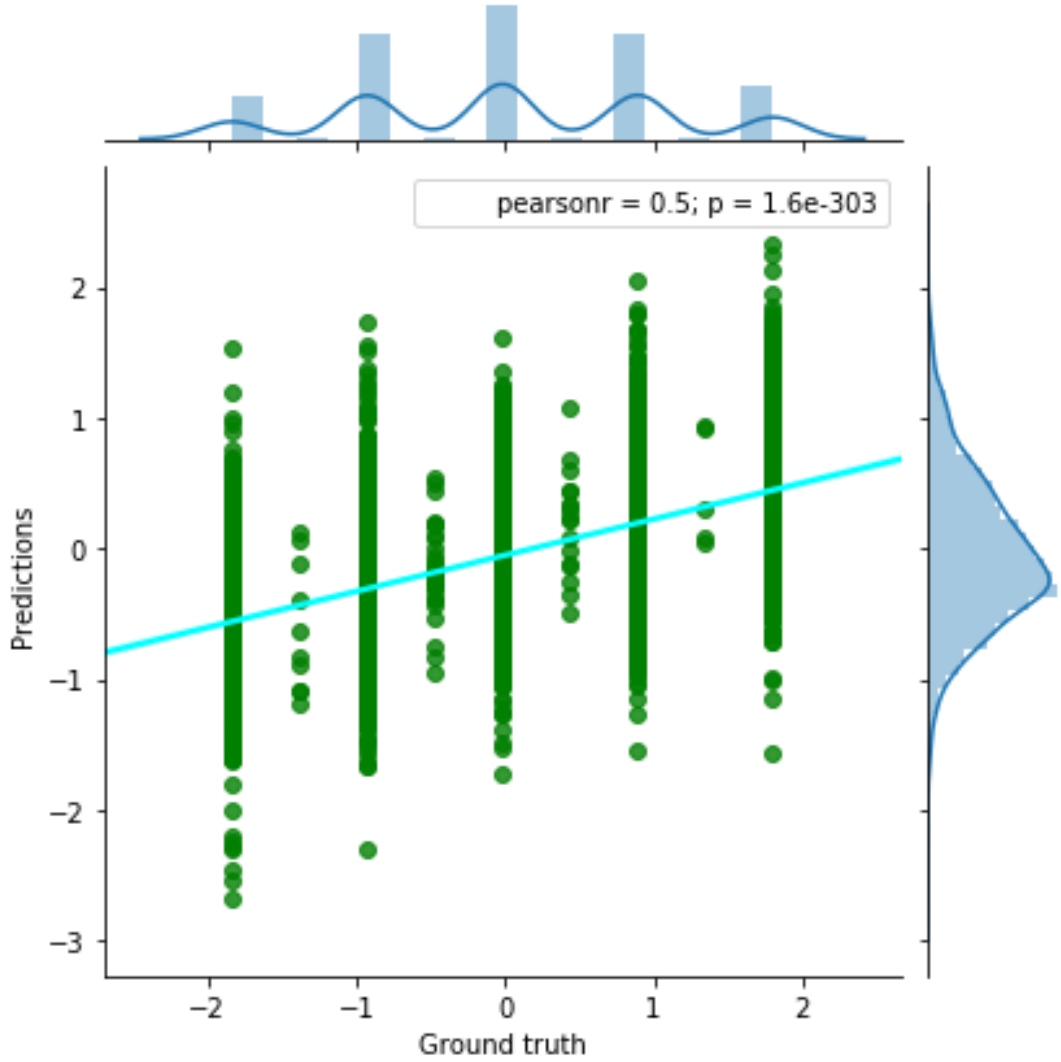


Figure 6.2: Relationship between predicted and real ratings of the baseline model. While it is true that there is some correlation between predicted and real ratings, there is still a lot of variance of the ratings that is not predicted by the baseline network. Nevertheless, with a test of hypothesis, we can see that there is a linear relationship, even if it is small, between real and predicted ratings $p \approx 0$.

These results are computed considering at the same time all the ratings in the test dataset. Consequently, there are users whose ratings may be predicted better than

others. This would mean that some of the workers in the test set have a behaviour that is considerably different to the workers in the test set, whereas others show a more predictable behaviour. This is what is indeed happening: The network shows considerably better results on the ratings on some workers than on other workers. In fact, if we compute the Spearman's ρ over the prediction of each user in the test dataset and then average them, we obtain a Spearman's ρ of 0.531, which is greater than when considering all ratings as a whole. MSE decreases slightly under this configuration, to 0.818.

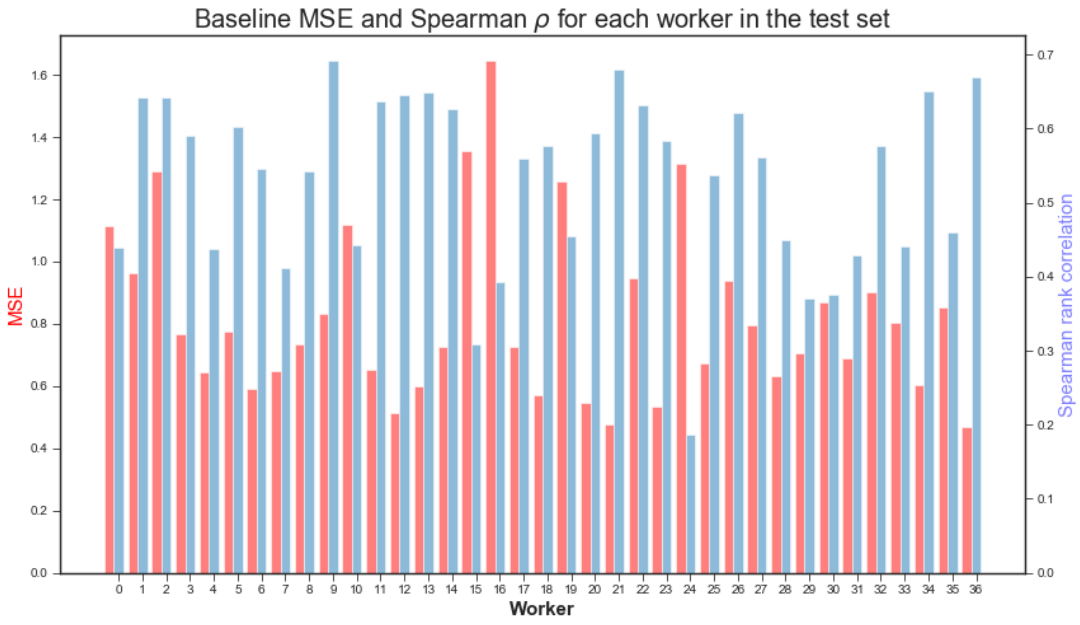


Figure 6.3: MSE and Spearman's ρ for each worker using the baseline model. It can be seen that there are workers for whom the MSE is exceptionally high (e.g. worker 16) and others for whom the MSE is exceptionally low (e.g. worker 36). A similar trend can be seen in Spearman's correlation. This shows that there are users who have very similar preferences to the users in the training set, and others that are significantly different.

There is an obvious negative correlation between the MSE and Spearman's ρ of the workers in the test dataset. The greater the mean squared error of the predictions, the lower the correlation rank. Nevertheless, this relationship is not perfectly linear, as it can be seen in the following plot. This shows that MSE, even if it can work as a loss function, does not perfectly predict the correlation between predicted and real ratings.

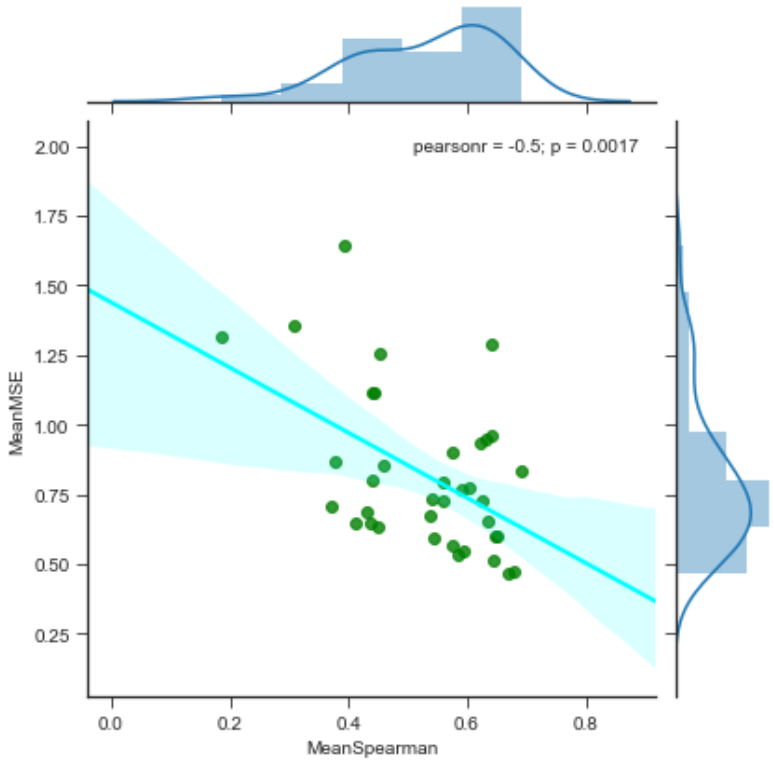


Figure 6.4: Relationship between Spearman's ρ and MSE of each worker using the baseline model.

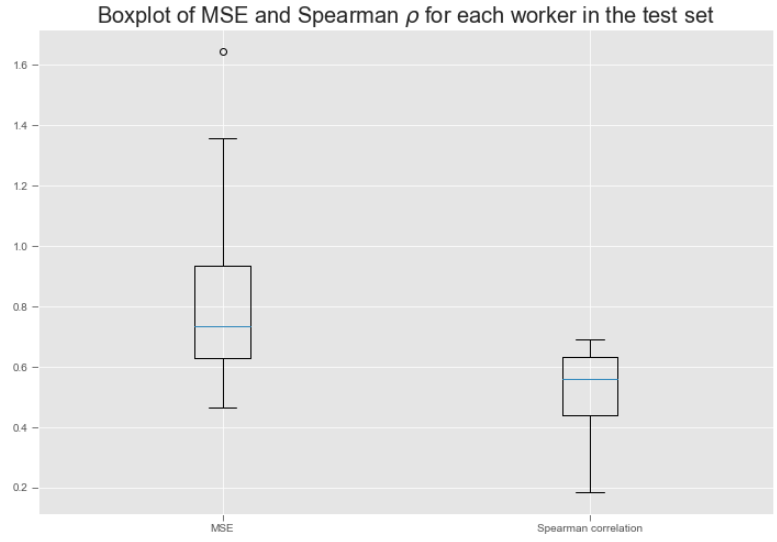


Figure 6.5: Boxplot of Spearman's ρ and MSE of each worker using the baseline model. It can be seen that the range of Spearman's rank correlation of each worker is of $[0.186, 0.692]$. This means that model performs more than 3 times better for some workers than for others.

6.2.1 What features did the mean aesthetic predictor network learn?

Neural networks are typically regarded as *black box* models, as their outputs cannot be explained. Nevertheless, in image classification, there are methods that can be used to understand what parts of the picture are more important for the network when it is performing classification or regression. One way to do this is using saliency maps (Simonyan et al., 2014; Shimoda and Yanai, 2016), which are typically used for image segmentation. They highlight the pixels of the picture that are more representative of a given class. As our problem is a regression problem, we can understand saliency maps in this context as representing the areas of the pictures that the network is using the most to understand their aesthetic value. Below, some examples taken from the Flickr-AES dataset are shown. They were obtained using SmoothGrad (Smilkov et al., 2017) as a way of reducing the noise of the saliency maps and adapting code found in a CNN visualization code repository[1].

Saliency maps

Figure 6.6: Some pictures obtained from the Flickr-AES dataset and their corresponding (at their right) saliency maps obtained using Smoothgrad. The pixels with the highest intensity of yellow show the areas of the picture that the network is using to construct the features that discriminate the aesthetic value of each picture. We can see that the network is focusing on the main elements of the pictures (flower, bird, etc.) and not giving as much importance to the background of the pictures.

[1] https://github.com/sar-gupta/convisualize_nb/blob/master/cnn-visualize.ipynb

Automatic enhancement of the pictures

When using CNNs, it is possible to obtain the gradient of the loss function with respect to the input image. This is the basis on which adversarial attacks are built; by performing small perturbations on the input pictures so that the network is deceived and outputs a wrong class (Szegedy et al., 2013). This property can also be used for understanding which features the network is looking for when assessing the aesthetic properties of a picture. By adding this gradient to the input picture, *enhanced* pictures can be obtained, which would have a greater predicted aesthetic value than the original picture.

This is used by the Fast Gradient Sign method (Gomes, 2018), that computes the new image as $X_{enhanced} = X_{original} + \varepsilon \times sign(\nabla_x J(X,Y))$, where ε controls the intensity of the change of the picture. Some examples on the use of this method for the automatic enhancement of pictures were obtained using our baseline model, in some pictures of the Flickr-AES dataset, which can be seen in the following plots. The code used to implement this method was adapted from a repository [2]. The perturbation image is the value of $sign(\nabla_x J(X,Y))$ and the value of ε was set to 0.2. It can be seen that, when adding the perturbations to the input image, the resulting image has more detail and their contrast is enhanced in some parts of the network, whereas in others, other changes are performed. The expected rating of all the enhanced pictures is greater than the original pictures.

This method, apart from being useful for understanding which features the network is looking for, creates the opportunity of using neural networks for intelligent photography aesthetics enhancement. If combined with a network that can take into account individual preferences over aesthetics in photography, this gradient ascent method can be used for personalized enhancement of pictures. It is a computationally cheap method, and it could be implemented in smart-phones or other computationally limited devices. It has limitations, as the contrasts and colours of a picture can be modified to a certain point, but it cannot modify characteristics like composition or style, which are also related to the aesthetic value of a picture.

[2] `https://savan77.github.io/blog/imagenet_adv_examples.html`

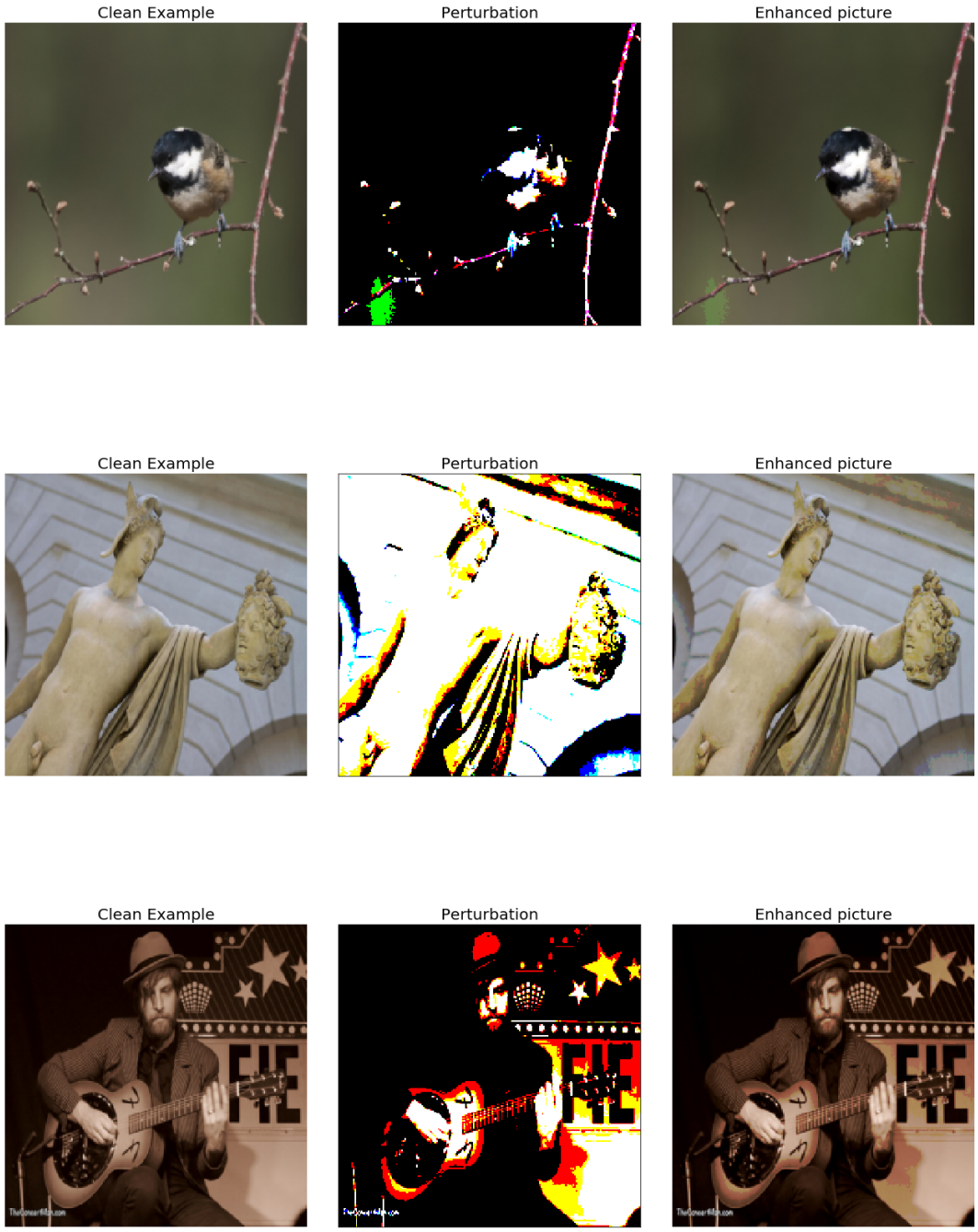


Figure 6.7: Enhanced images using Fast Gradient Sign. The pictures were obtained from the Flickr-AES dataset. All the enhanced images had a greater expected rating (around 10% bigger) than their corresponding original pictures.

6.3 Personalised aesthetic prediction networks

6.3.1 10 images available for each user

The 10-fold cross validation results were as follows: The Spearman's ρ was increased, in average, by 0.044, and MSE was decreased, in average, by 0.248. The standard deviations of the changes of both metrics were, respectively: 0.018 and 0.129. The minimum increase in Spearman's ρ was of 0.021 and the maximum was of 0.120. The increase in Spearman's ρ for each worker can be visualized in the following plot:

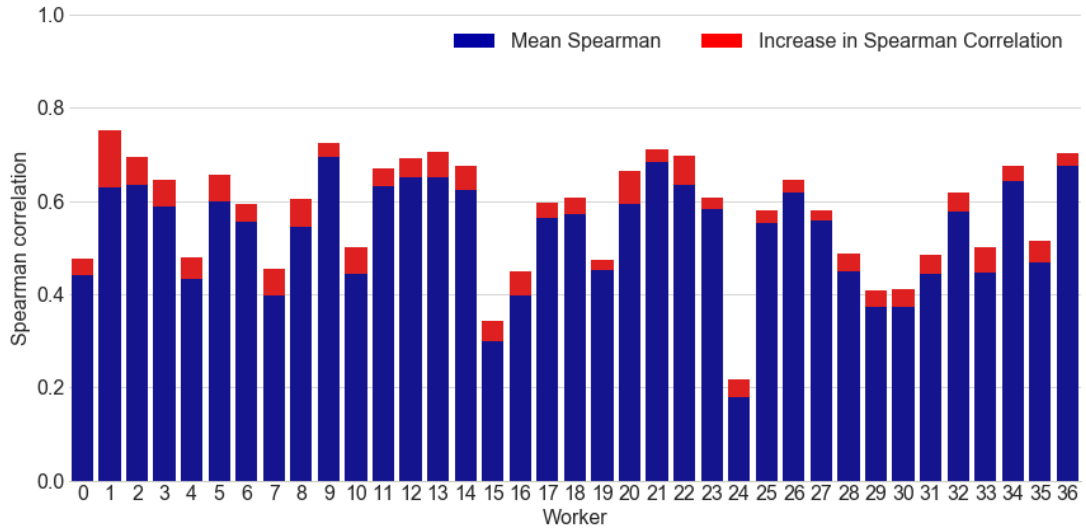


Figure 6.8: Increase in Spearman's ρ in the k=10 experiment configuration. The blue bars show the mean Spearman's correlation of the pictures that were not used to fine-tune the model, and the red bars show the average increase for this correlation in the 10-fold experiment.

In the plot above, it can be seen that, when the model can observe 10 ratings of new pictures for each user, it can increase its prediction capabilities, in average, by 0.044. This increase in performance does not depend on the accuracy of the baseline model, as it is quite uniform across all workers. This relationship can be visualized better in the following plots:

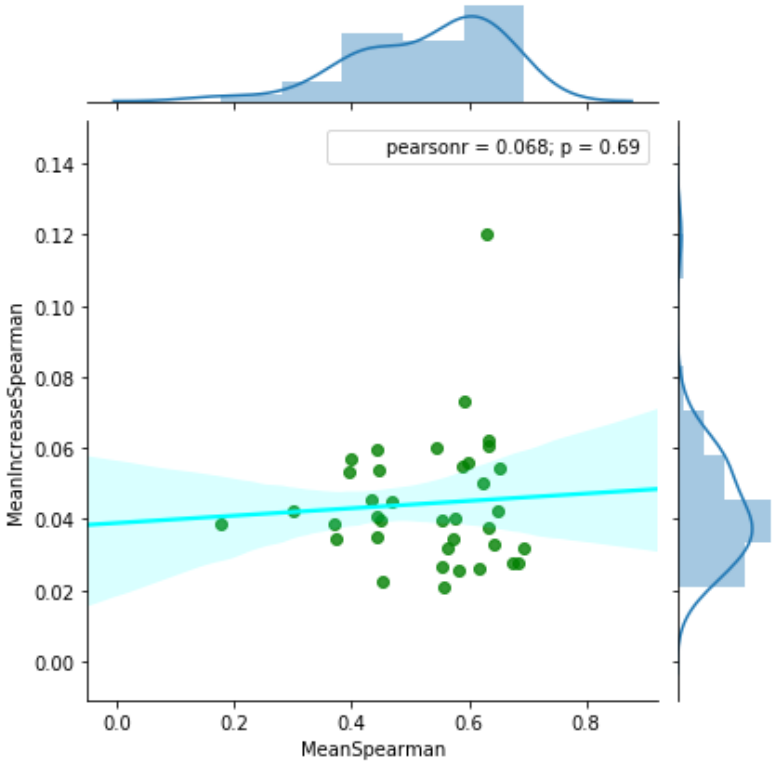


Figure 6.9: Relationship between the increase in Spearman's ρ and the baseline's correlation. Both variables are not linearly dependent $p = 0.69$

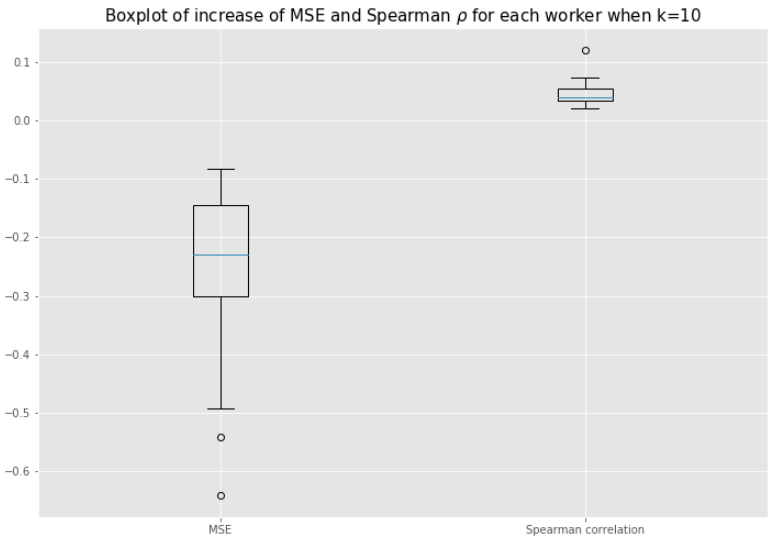


Figure 6.10: Boxplot of increase Spearman's ρ and MSE of each worker using the k=10 configuration. The range of increases of the Spearman's ρ was considerably uniform, in the range of $[0.021, 0.120]$, whereas the change in MSE showed a greater variance.

6.3.2 100 images available for each user

Fine-tuning the last block of layers

The 3-fold cross validation results were as follows: The Spearman's ρ was increased, in average, by 0.067, and MSE was decreased, in average, by 0.148. The standard deviations of the changes of both metrics were, respectively: 0.049 and 0.107. The minimum increase in Spearman's ρ was of 0.001 and the maximum was of 0.166. The increase in Spearman's ρ for each worker can be visualized in the following plot:

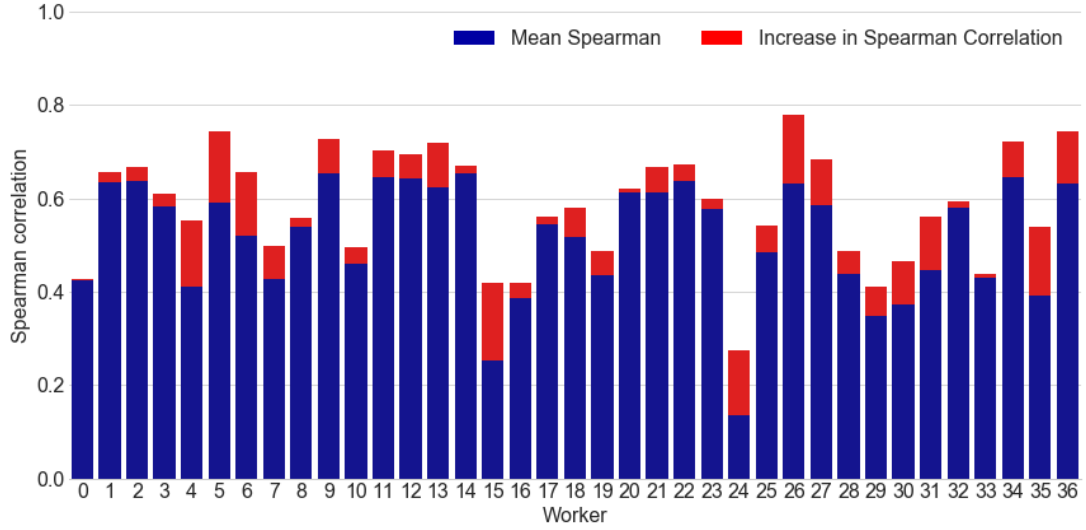


Figure 6.11: Increase in Spearman's ρ in the k=100 experiment configuration, when fine tuning only the last layers. The blue bars show the mean Spearman's correlation of the pictures that were not used to fine-tune the model, and the red bars show the average increase for this correlation in the 3-fold experiment.

In the plot above, it can be seen that there are workers for which the model have increased significantly its predictive capacity, whereas for others this increase has not been as large. For example, for workers 24 and 26, the increase has been significant, but the model was not able to learn much from workers 0 and 33 by finetuning only the last layers. There is some negative correlation between the baseline's prediction performance and the increase of performance, as it can be seen in the following plot:

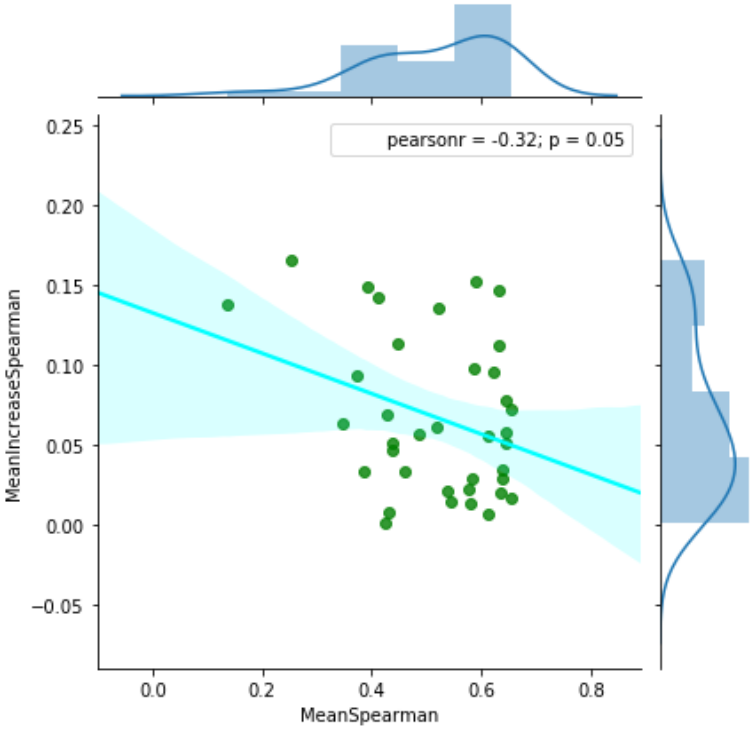


Figure 6.12: Relationship between the increase in Spearman's ρ and the baseline's correlation in the k=100 experiment configuration, when fine tuning only the last layers. Both variables may be linearly dependent $p = 0.05$

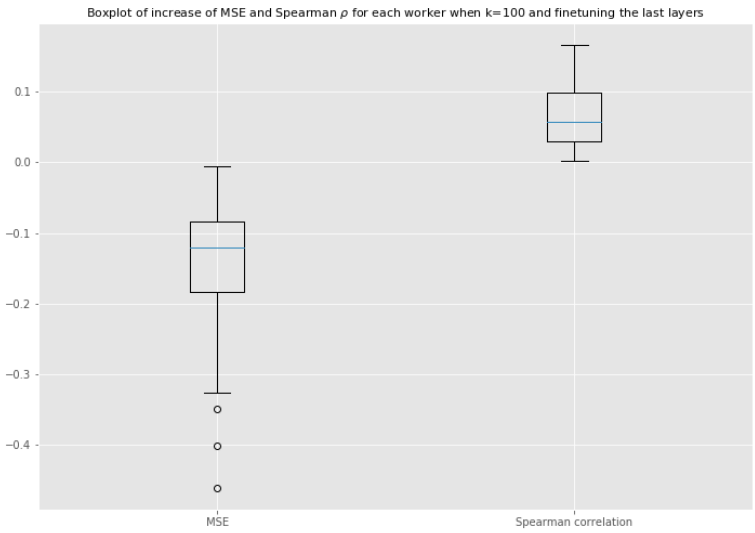


Figure 6.13: Boxplot of increase Spearman's ρ and MSE of each worker using the k=100 configuration, when fine tuning only the last layers. The range of increases of the Spearman's ρ was less uniform, in the range of $[0.001, 0.166]$, than in the k=10 setting.

Fine-tuning all the layers in the network

The 3-fold cross validation results were as follows: The Spearman's ρ was increased, in average, by 0.115, and MSE was decreased, in average, by 0.165. The standard deviations of the changes of both metrics were, respectively: 0.095 and 0.134. The minimum increase in Spearman's ρ was of 0.007 and the maximum was of 0.480. The increase in Spearman's ρ for each worker can be visualized in the following plot:

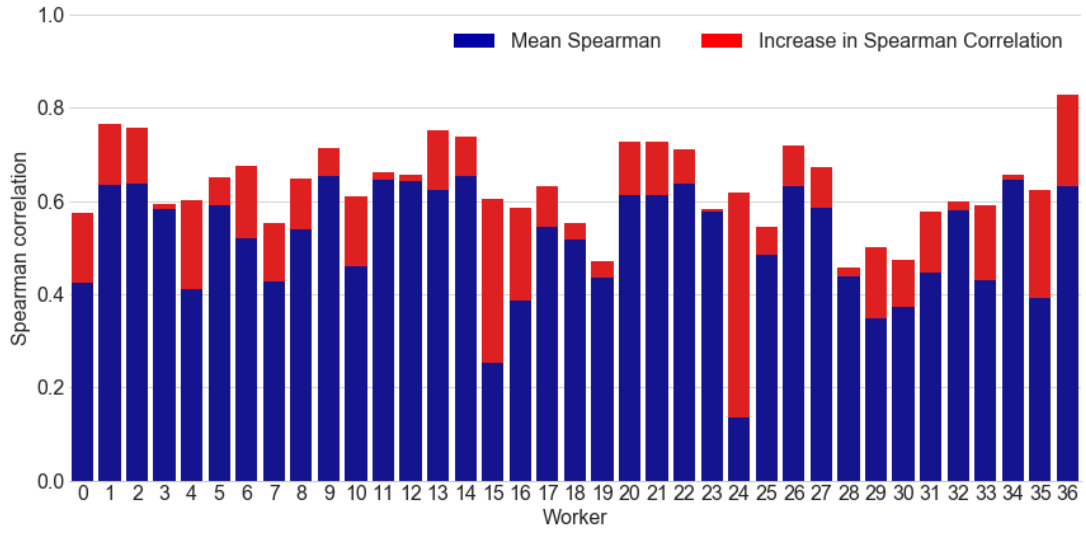


Figure 6.14: Increase in Spearman's ρ in the k=100 experiment configuration, when fine tuning every layer. The blue bars show the mean Spearman's correlation of the pictures that were not used to fine-tune the model, and the red bars show the average increase for this correlation in the 3-fold experiment.

In the plot above, it can be seen that the model, when fine-tuning every layer, can significantly increase its performance for almost every worker. This increase in performance is negatively related to the performance of the baseline model. We argue that this negative relationship indicates that there are some workers (e.g. workers 15 and 24) who showed significantly different behaviour to the workers in the training set, but when using some fine-tuning methods, their preferences can be learned by adjusting the features learned in the training set. There is, consequently, an effect of balancing the performance across all workers, as the workers for whom the baseline network showed less predictive capacity were the workers for whom the fine-tuned network showed a greater increase in predictive capacity, and viceversa. This negative correlation can be seen better in the following plot:

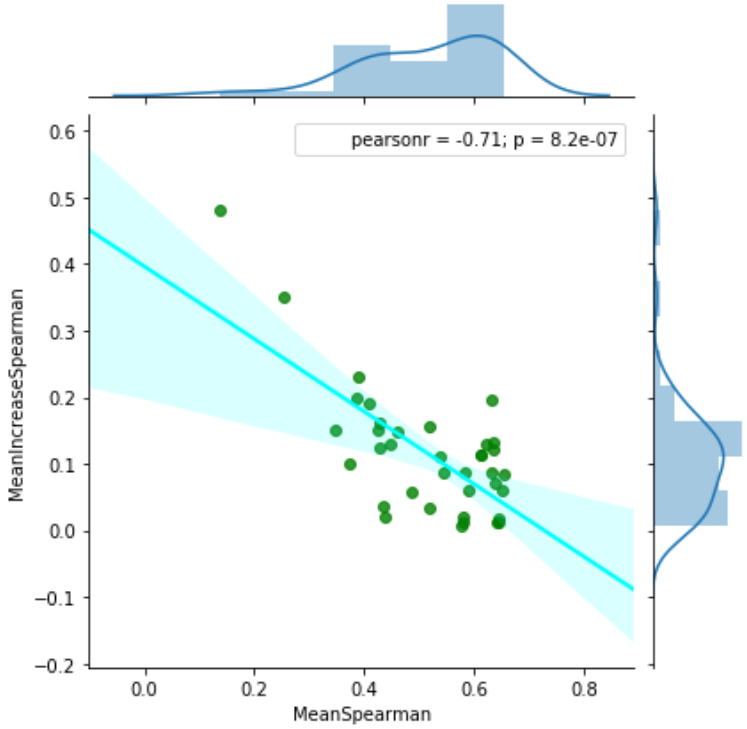


Figure 6.15: Relationship between the increase in Spearman's ρ and the baseline's correlation in the k=100 experiment configuration, when fine tuning all layers. Both variables are likely to be linearly dependent $p \approx 0$

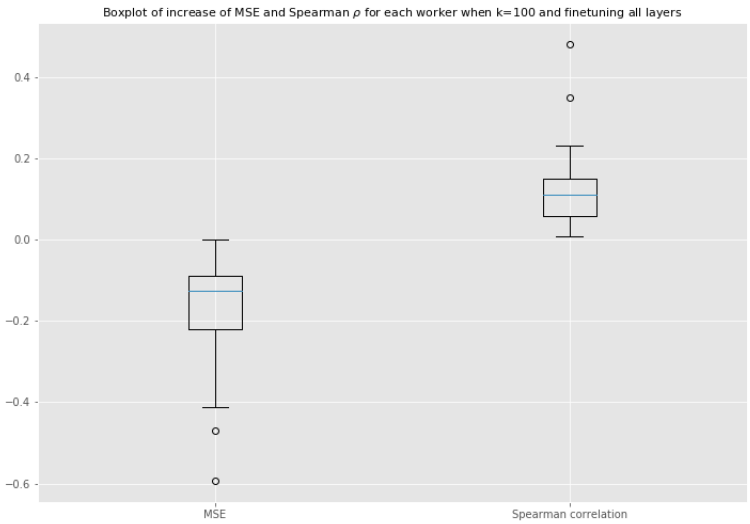


Figure 6.16: Boxplot of increase Spearman's ρ and MSE of each worker using the k=100 configuration, when fine tuning every layer. The range of increases of the Spearman's ρ was of $[0.007, 0.480]$.

Experiments with residual adapters

The 3-fold cross validation results of the network with adapters in parallel to every 3x3 convolutional layer were as follows: The Spearman's ρ was increased, in average, by 0.122, and MSE was decreased, in average, by 0.192. The standard deviations of the changes of both metrics were, respectively: 0.150 and 0.236. The minimum increase in Spearman's ρ was of 0.0 and the maximum was of 0.751. The increase in Spearman's ρ for each worker can be visualized in the following plot:

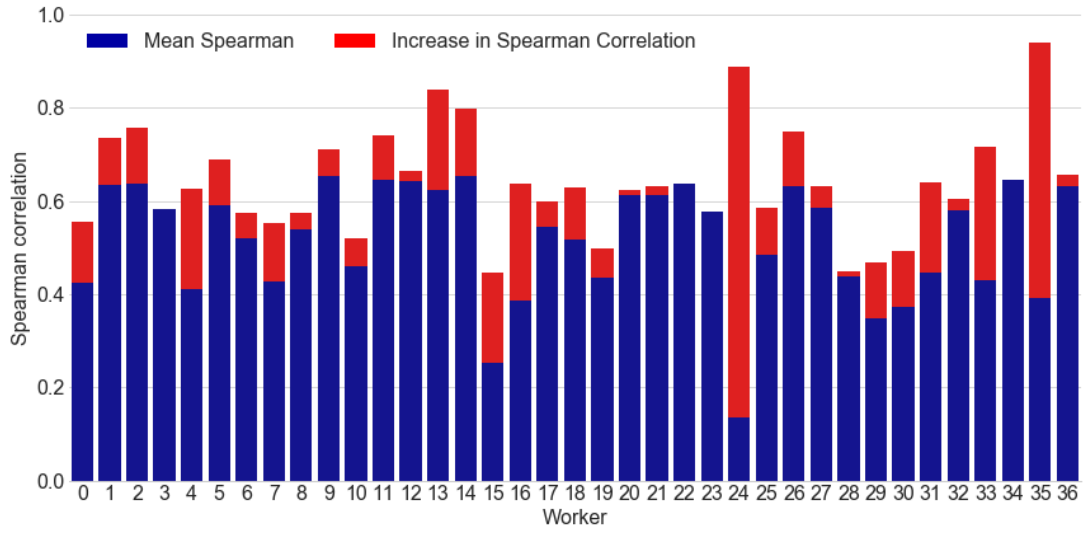


Figure 6.17: Increase in Spearman's ρ in the k=100 experiment configuration, when adding the adapters in parallel to every 3x3 convolutional layer.

In the plot above, it can be seen that the performance of the network with the adapters shows a greater variance than the network for which every layer was fine-tuned. Some of the workers for which the baseline model showed a worse performance were the ones that the new network could learn the most from. In particular, workers 24 and 35 had a significant increase in performance. Interestingly, there are two workers (34 and 3) for whom the performance did not increase whatsoever compared to the baseline model, thus showing some degree of overfitting. The balancing effect that was seen when fine-tuning every layer is not as significant in this experiment. Nevertheless, some negative correlation still exists, as it can be seen in the following plot:

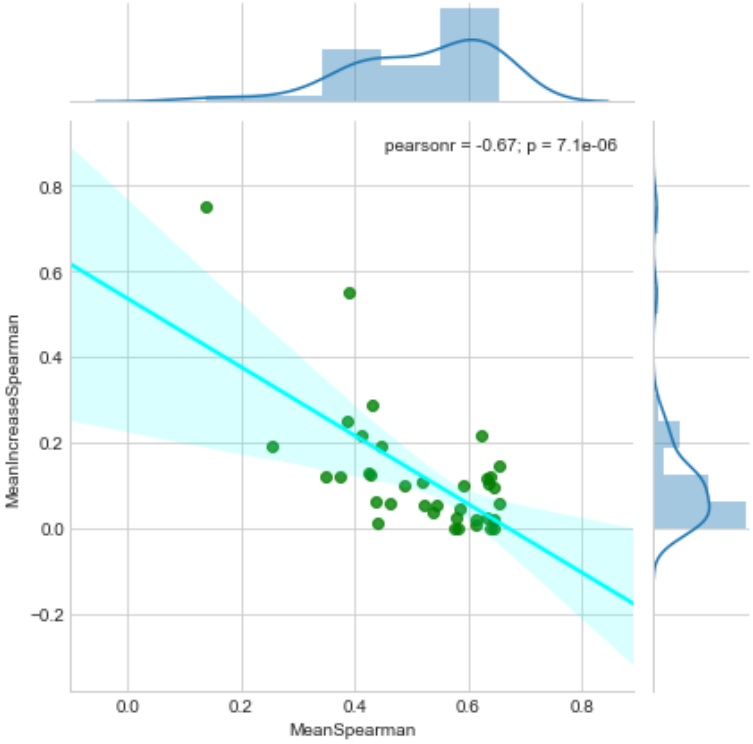


Figure 6.18: Relationship between the increase in Spearman's ρ and the baseline's correlation in the k=100 experiment configuration, when when using adapters. Both variables are likely to be linearly dependent $p \approx 0$

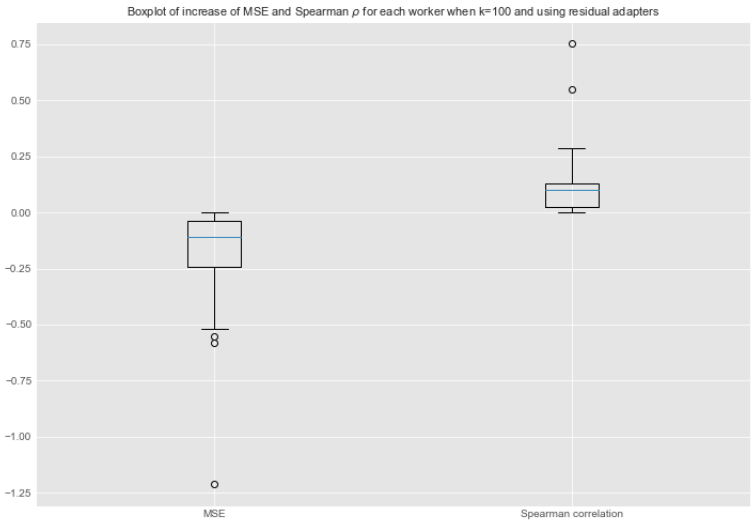


Figure 6.19: Boxplot of increase Spearman's ρ and MSE of each worker using the k=100 configuration, when using adapters. The range of increases of the Spearman's ρ was less uniform, in the range of $[0.0, 0.751]$, than in the k=100 and fine-tuning all layers setting.

Interestingly, the decrease in MSE and the increase in Spearman's rank are not perfectly correlated, as it can be seen in the plot below.

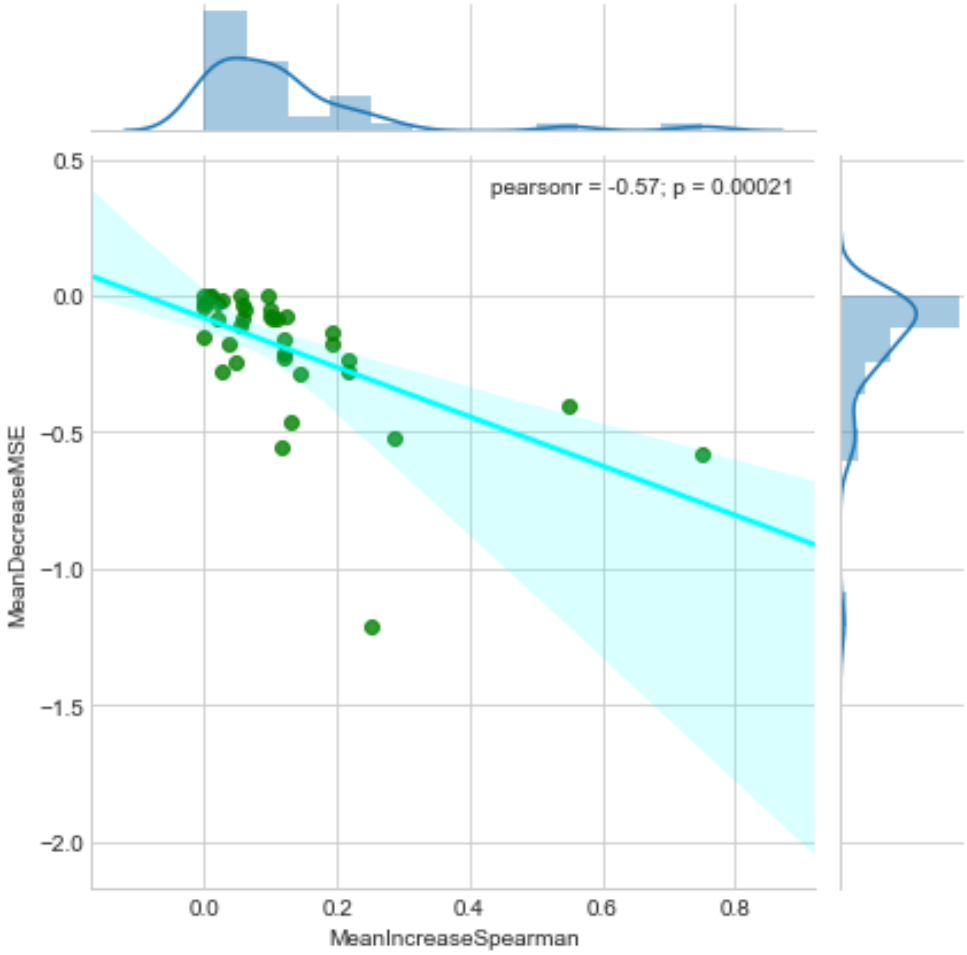


Figure 6.20: Relationship between the increase in Spearman's ρ and MSE of each worker using the k=100 configuration, when using adapters. The two variables are probably related $p < 0.001$, but the Pearson's correlation is of only 0.57, which suggest that decreasing the mean squared error does not necessarily translate into an increase in Spearman's ρ of the same magnitude, or viceversa.

In the experiment where the adapters were positioned in the last 2 blocks of convolutional layers, the mean increase in spearman's ρ was of 0.081642, and when they were positioned in the last 3 blocks of layers, the mean increase was of 0.1205, with a standard deviation of 0.141. Those results are aligned to what was reported in (Rebuffi et al., 2018), where they found that using adapters throughout all the 3x3 convolutional layers in the network worked better than positioning them in parallel to only a subset of those layers.

6.4 Method comparison

The following table summarises the results of the best-performing models that were found for each set of experiments:

Model	Mean ρ	σ(ρ)	Median ρ	Min. ρ	Max. ρ
Baseline	0.531	0.118	0.561	0.186	0.692
K=10	0.575	0.122	0.604	0.217	0.750
K=100 last layers	0.584	0.118	0.593	0.274	0.778
K=100 all layers	0.632	0.088	0.623	**0.458**	0.828
K=100, adapters late	0.637	0.112	0.623	0.436	0.937
K=100, adapters all	**0.639**	0.115	**0.631**	0.445	**0.941**

Table 6.1: Comparison of the performance on unseen data of the methods studied. It can be seen that the method that shows a greater capacity of predicting the ratings of unseen pictures is the method that included the adapters in parallel to each 3x3 convolutional layer in the network, as it shows the greatest mean and median. The minimum and maximum correlation obtained for the 37 workers in the test set are also reported, as well as the standard deviation (σ). Interestingly, the method that showed a greatest minimum value was the method for which all the parameters were fine tuned. The network with residual adapters in the later 3 blocks of layers showed a very similar performance to the network with adapters throughout every layer.

Interestingly, by performing a one-tailed test of hypothesis to compare the distributions of the Spearman's ρ for each method using the Welch's t-test (Ahad and Yahaya, 2014), we obtain that the three last methods in the table below have shown results that are not significantly different ($p > 0.5$) when comparing each of the three methods with the other two.

Those distributions can be seen with more detail in the following plot, which in fact shows that those 3 last methods show very similar results.

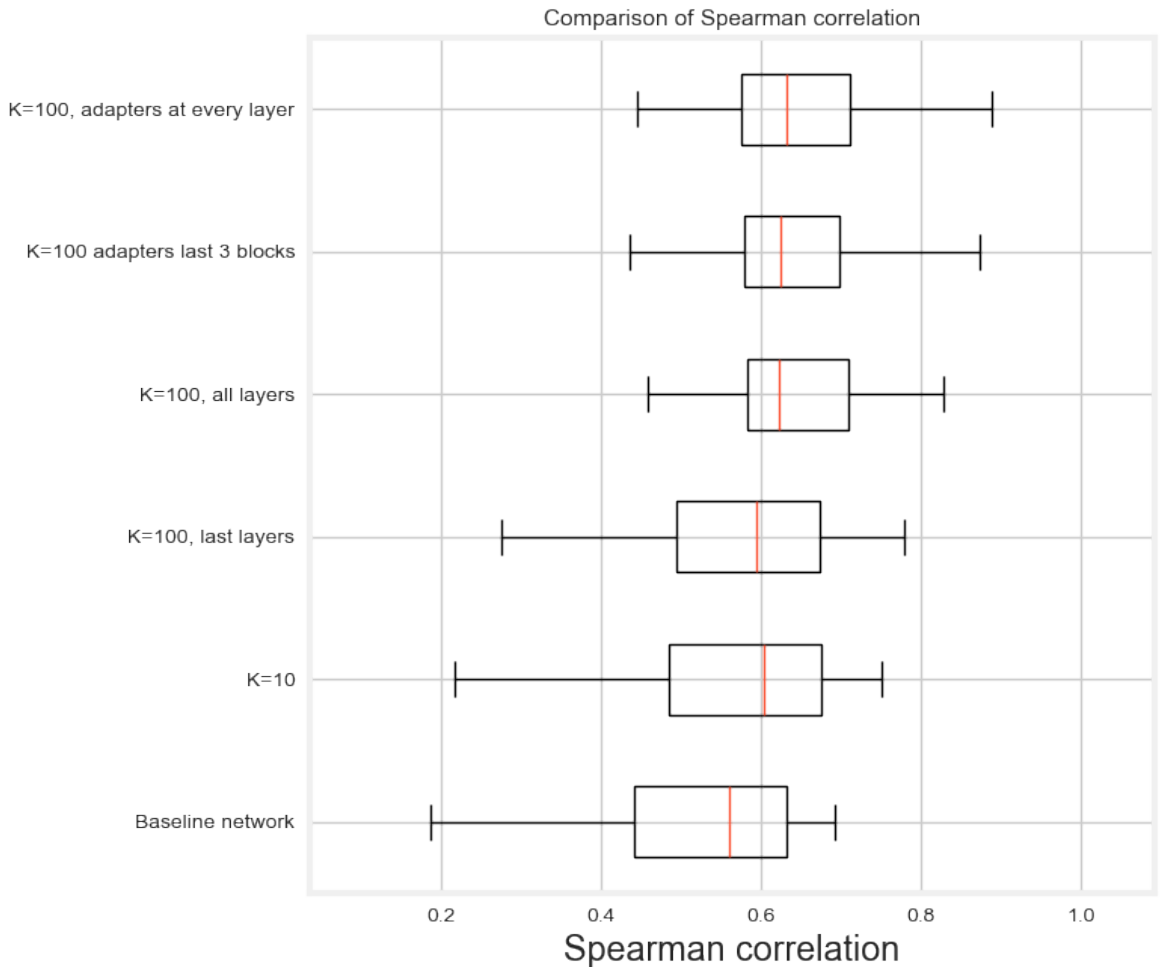


Figure 6.21: Boxplot comparing the results of each method on unseen pictures for the 37 workers in the test dataset, in terms of Spearman's ρ.

Chapter 7

Conclusion

7.1 Main findings

Throughout this project, it has been found that there are methods capable of understanding personal preferences on aesthetics in photography that do not require information about the contents and styles of pictures, and which can improve the state-of-the-art results on a particular dataset. The main addition in this project is the proposed residual adapter architecture, which obtains the best results of every model that was implemented while also exploiting parameter re-use, thus reducing the need for end-to-end user-specific models that would require a considerable amount of computational resources. The findings in the literature regarding the position of those adapters were also validated. By utilising different methods using the model proposed in this thesis, we were also able to visualize the parts of the picture that the network considers more important for the aesthetics component. It is hypothesised that some of those methods can also be used for personalised picture enhancement.

It was found that the prediction capabilities of our model depend in a significant way on the user, as there are users who show preferences that are more aligned with the users in the training set than others. Nevertheless, it was also found that the parameters (filters) that explain the preferences of the users in the training dataset are, up to a point, reusable (with some adjustments) for new, unseen, users. This shows that the perception of beauty in photography is not completely subjective, and some of the individual preferences can be predicted by machine learning models if given enough data. These results were obtained without using additional information that could increase our understanding on aesthetics (e.g. the situation surrounding the user when seeing the picture, the medium used to visualize the picture, its contents, style, etc.).

7.2 Future work

7.2.1 Improvements on our experiments

Research on the baseline model

To construct the baseline mean aesthetics prediction model, a pre-trained Resnet-18 on the training set of the FLICKR-AES dataset was fine-tuned. This was done because it was found that deeper models did not improve the generalization capabilities of the model, and were harder to train. Nevertheless, it is possible that this baseline model can be improved. A possible solution could be to train the model on another, larger, aesthetics-related dataset, like the AVA dataset, and fine-tune it on the FLICKR-AES dataset. Our approach involved fine-tuning filters that were learned for image classification tasks, not for aesthetics. The use of inception modules, which have various filter sizes, could also be attempted, as it is possible that there are aesthetic features which need larger filter sizes to be understood. Research in CNNs and deep learning is constantly finding new, improved ways of training deep models, and it is likely that our baseline model can be improved in several ways (network architectures, different transfer learning methods, learning rules, activation functions, data augmentation methods, etc.). This baseline network learns all the features that are reused by the user-specific networks in our approach, which is why it is important to have the best possible baseline model, with a significant variety of features that should be representative of as many aesthetic-related patterns as possible.

Further research on the user-specific models

Even if it was found that the best model involved adding adapters to each 3x3 convolutional layer in the baseline network, it was also found that this method shows a greater variance in performance than fine-tuning all the layers in the network. If used in a real-world system, this would mean that the network with the adapters would work worse for some users than the network where all the layers were fine-tuned. Further experiments could be performed to try to solve this problem, like performing a more careful training (better hyper-parameter selection), testing other data augmentation methods, etc. It was argued that the adapters were preferable in situations where computational resources are limited, as they favour parameter-reuse. Further work could be performed to reduce even more the impact of those adapters in the model size. The implementation of those adapters was made in a way that allowed the exper-

iments to be run quickly, but this implementation could be considerably improved for its use on real-world systems.

Most of the experiments were run in a 3-fold cross-validation fashion due to the computationally-extensive nature of the problem. Increasing the number of folds would lead to a more fair estimate of the results of each model, thus allowing us to draw more definitive conclusions.

Use of data augmentation

It was argued earlier in this document that there were data augmentation methods that are not a good fit for the training of aesthetic prediction models. The kind of data augmentation that is used limits the type of features that the network looks for in the images. For example, if random cropping is used, the network will not be looking for features like symmetry, as this kind of feature is eliminated when randomly choosing a subsection of the image. Consequently, it is possible that, if more data was available, this kind of data augmentation method could be eliminated, so that the model can look for composition-related features, thus being able to perform richer predictions. The fact that this kind of data augmentation was used is an important limitation of our methodology. The use of generative adversarial networks for data augmentation was not attempted in this project, as it was considered that the generation of fake images would require further human annotation, which is something that we wanted to avoid. Moreover, training the generative network would take a considerable amount of research time that could be used for other experiments. The use of GANs can, nevertheless, improve the baseline or the user-specific networks performances, and it would be interesting to do research on this topic.

Reducing the limitations of the models

Another limitation of our proposed methodology was the fact that the network always receives squared (224x224 pixels) images. This requires that every non-squared image is changed to a different proportion (thus distorting the composition of the picture, sizes of the objects, relative distances, perspectives, etc.), and downsized (thus reducing the quality of the pictures themselves). This preprocessing on the pictures greatly limits the kind of features that the model can find on the pictures, such as features related to composition, pixel quality, textures, etc. Further research could be perform to remove this requirement partially or totally. It is possible that, if the network accepts

inputs with different sizes, its performance can increase, but it is also possible that the baseline model has to be significantly different, and the experiments performed in this thesis should be repeated, to confirm or discard our results in this different setting. Moreover, it is possible that the resolution of the pictures played an important role when the workers assigned a score to each picture. This information is lost when resizing the images and, consequently, the model is not able to detect this pattern. Additionally, the network does not accept pictures with a channel of transparency, but this is something that is unlikely in photography due to the nature of digital cameras.

7.2.2 Further experiments

Research on recommender systems

The methodology proposed in this system could be translated into recommender systems. It was argued in the introduction that there are situations in which collaborative filtering, which is the basis for many recommender systems, could not be used. For example, when the set of images that each user rates does not overlap with the set of images that other users rated in the past (e.g., personal albums). This kind of situation requires that the recommender system does not take into account similarity between users by checking how similar their preferences were in the past. Our proposed methodology fits in this kind of situation, as, after the baseline model is trained, it does not require to learn from other users to perform prediction on the preferences on one user. Nevertheless, once the model learns from one user, the methodology could be re-used, by adding those learned preferences to the baseline model. Research could be performed to find an optimal way to design this kind of recommender system with the transfer learning methodology that was used. This research could also find improvements to our methodology. Moreover, if this methodology was applied in a Context-Aware recommender system (Adomavicius and Tuzhilin, 2015), it would also be possible to increase the overall performance of the model by taking into account its predictions (which, in average, explain a 63% of the preferences of the users), as well as contextual data (what is the user doing, what are their surroundings, what kind of medium is being used to visualize the picture, or even mood), to further adjust the predictions of the system.

Research on personalised enhancement of photographs

It was briefly demonstrated in this thesis that the baseline network could be used to slightly improve the aesthetic appearance of pictures by performing a gradient ascent operation on them. We argue that this opens the opportunity for the creation of user-specific picture filters, that depend on the past preferences of each user. For example, some users may prefer a greater blur of the background than others, and the network may be able to distinguish those preferences. Research could be performed to test this hypothesis, and to check its viability in real-world settings, like in automatic personalised image enhancements in smart-phones or social networks. This kind of experiment may need human intervention to find the best kind of gradient-ascent method to use, hyper-parameters, etc.

Research on different datasets

The results obtained in this project were only tested in the FLICKR-AES dataset, as it is the largest dataset available in which we can distinguish who gave each rating. The results obtained are thus limited to this dataset. How representative this dataset may be of the problem is questionable, as the kind of people that rated the pictures may not be representative of the whole population, and the pictures present in them may not be a good representation of the pictures that would be found in a real-world setting. For instance, it is possible that, if we were to include our method in a domain where only landscape pictures are present (e.g., a landscape-specific photography repository) , training the model on the whole FLICKR-AES dataset may not be that appropriate, as the model would also learn features that are not related to aesthetics in landscapes. The level in which the features that explain aesthetics in one domain can be exploited in other domains can be limited.

If new datasets were made publicly available, with the ratings of many more users of more pictures, it should be possible to find more patterns that explain the preferences of the users, which would lead to more generalizable filters in the networks. Consequently, it is more likely that the preferences of unseen users can be explained by the users in the dataset used to train the model, as it should be more representative of the overall set of potential users of a photography-rating system.

Bibliography

ImageNet.

(2016). Image File Types: Top 5 Types Of Picture Formats.

(2018). CS231n Convolutional Neural Networks for Visual Recognition.

Adams, R. (1982). *Beauty in photography : essays in defense of traditional values*.

Adomavicius, G. and Tuzhilin, A. (2015). Context-Aware Recommender Systems. In *Recommender Systems Handbook*, pages 191–226. Springer US, Boston, MA.

Agrawal, A. (2017). Loss Functions and Optimization Algorithms. Demystified.

Ahad, N. A. and Yahaya, S. S. S. (2014). Sensitivity analysis of Welch's t-test. In *AIP Conference Proceedings*, volume 1605, pages 888–893. American Institute of Physics.

Alif, M. A. R., Ahmed, S., and Hasan, M. A. (2017). Isolated Bangla handwritten character recognition with convolutional neural network. In *2017 20th International Conference of Computer and Information Technology (ICCIT)*, pages 1–6. IEEE.

Antoniou, A., Storkey, A., and Edwards, H. (2018). Data augmentation Generative Adversarial Networks. Technical report.

Appu Shaji (2016). Understanding Aesthetics with Deep Learning.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer Normalization.

Benjamin, W. (1935). *The Work of Art in the Age of Mechanical Reproduction*. Illuminations.

Bhattacharya, S., Sukthankar, R., and Shah, M. (2010). A framework for photo-quality assessment and enhancement based on visual aesthetics. In *Proceedings of the international conference on Multimedia - MM '10*, page 271, New York, New York, USA. ACM Press.

Bianco, S., Celona, L., Napoletano, P., and Schettini, R. (2016). Predicting Image Aesthetics with Deep Learning. pages 117–125. Springer, Cham.

Browniee, J. (2017). A Gentle Introduction to Transfer Learning for Deep Learning.

Chen, C., Chen, U. Q., Labs, I., and Xu, J. (2018). Learning to See in the Dark.

Chen, C., Seff, A., Kornhauser, A., and Xiao, J. (2016). DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving.

Chen, Y. L., Huang, T. W., Chang, K. H., Tsai, Y. C., Chen, H. T., and Chen, B. Y. (2017). Quantitative analysis of automatic image cropping algorithms: A dataset and comparative study. *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, pages 226–234.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2016). FAST AND ACCURATE DEEP NETWORK LEARNING BY EXPONENTIAL LINEAR UNITS (ELUS).

Cybenkot, G. (1989). Mathematics of Control, Signals, and Systems Approximation by Superpositions of a Sigmoidal Function*. *Math. Control Signals Systems*, 2:303–314.

Datta, R., Joshi, D., Li, J., and Wang, J. Z. (2006). Studying aesthetics in photographic images using a computational approach. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3953 LNCS, pages 288–301. Springer, Berlin, Heidelberg.

Datta, R., Li, J., and Wang, J. Z. (2008). Algorithmic inferencing of aesthetics and emotion in natural images: An exposition.

Deng, Y., Loy, C. C., and Tang, X. (2017). Image Aesthetic Assessment: An experimental survey. *IEEE Signal Processing Magazine*, 34(4):80–106.

Denzler, J., Rodner, E., and Simon, M. Convolutional Neural Networks as a Computational Model for the Underlying Processes of Aesthetics Perception.

Dietz, M. (2017). Understand Deep Residual Networks — a simple, modular learning framework that has redefined....

Dodge, S. and Karam, L. (2016). Understanding How Image Quality Affects Deep Neural Networks.

Dubovikov, K. (2017). PyTorch vs TensorFlow: spotting the difference.

Fang, H. and Meng Zhang, G. (2017). Creatism: A deep-learning photographer capable of creating professional work.

Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image Style Transfer Using Convolutional Neural Networks.

Gharbi, M., Csail, M., Chen, J., Research, G., Barron, J. T., Hasinoff, S. W., and Durand, F. (2017). Deep Bilateral Learning for Real-Time Image En-hancement. *ACM Trans. Graph*, 36.

Gomes, J. (2018). Adversarial Attacks and Defences for Convolutional Neural Networks.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Han, D., Kim, J., and Kim, J. (2017). Deep Pyramidal Residual Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6307–6315. IEEE.

Hannun, A. (2017). PyTorch or TensorFlow?

Hayn-Leichsenring, G. U., Lehmann, T., and Redies, C. (2017). Subjective Ratings of Beauty and Aesthetics: Correlations With Statistical Image Properties in Western Oil Paintings.

He, K. and Sun, J. (2015). Convolutional Neural Networks at Constrained Time Cost. *CVPR*, pages 5353–5360.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Identity Mappings in Deep Residual Networks. pages 630–645. Springer, Cham.

Hinton, G., Srivastava, N., and Swersky, K. (2012). Neural Networks for Machine Learning Lecture 6a Overview of mini--batch gradient descent.

Hong, L., Doumith, A. S., and Davison, B. D. (2013). Co-factorization machines. In *Proceedings of the sixth ACM international conference on Web search and data mining - WSDM '13*, page 557, New York, New York, USA. ACM Press.

Hubel, D. H. and Wiesel, A. T. N. (1962). Receptive Fields, Binocular Interaction and Functional Architecture in the Cat's Visual Cortex. *J. Physiol*, 160:106–124.

Inoue, H. (2018). Data Augmentation by Pairing Samples for Images Classification. Technical report.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.

Isinkaye, F., Folajimi, Y., and Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273.

Janocha, K. and Czarnecki, W. M. (2017). On Loss Functions for Deep Neural Networks in Classification.

Jia, Y. (2018). Caffe — Model Zoo.

Jiang, W., Loui, A. C., and Cerosaletti, C. D. (2010). Automatic aesthetic value assessment in photographic images. In *2010 IEEE International Conference on Multimedia and Expo*, pages 920–925. IEEE.

Jin, X., Chi, J., Peng, S., Tian, Y., Ye, C., and Li, X. (2016). Deep Image Aesthetics Classification using Inception Modules and Fine-tuning Connected Layer.

Jin, X., Zhao, M., Chen, X., Zhao, Q., and Zhu, S.-C. (2010). Learning Artistic Lighting Template from Portrait Photographs. pages 101–114. Springer, Berlin, Heidelberg.

Joshi, D., Datta, R., Fedorovskaya, E., Luong, Q.-T., Wang, J., Li, J., and Luo, J. (2011). Aesthetics and Emotions in Images. *IEEE Signal Processing Magazine*, 28(5):94–115.

Karpathy, A. (2018). Transfer Learning: CS231n Convolutional Neural Networks for Visual Recognition.

Khosla, A., Das Sarma, A., and Hamid, R. (2014). What makes an image popular? *Proceedings of the 23rd international conference on World wide web - WWW '14*, pages 867–876.

Kim, J., Kwon Lee, J., and Mu Lee, K. (2016). Accurate Image Super-Resolution Using Very Deep Convolutional Networks. *CVPR*, pages 1646–1654.

Kingma, D. P. and Lei Ba, J. (2014). Adam: A method for stochastic optimization.

Kong, S., Shen, X., Lin, Z., Mech, R., and Fowlkes, C. (2016). Photo Aesthetics Ranking Network with Attributes and Content Adaptation.

Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images.

Krizhevsky, A. (2010). Convolutional Deep Belief Networks on CIFAR-10.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks.

Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., and Navab, N. (2016). Deeper Depth Prediction with Fully Convolutional Residual Networks. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 239–248. IEEE.

Lange, S. and Riedmiller, M. (2010). Deep Auto-Encoder Neural Networks in Reinforcement Learning.

Lawrence, S., Giles, C., Ah Chung Tsoi, and Back, A. (1997). Face recognition: a convolutional neural-network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113.

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

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998a). Gradient-Base Learning Applied to Document Recognition.

LeCun, Y., Cortes, C., and Burges, C. (1998b). MNIST handwritten digit database,.

Leder, H., Belke, B., Oeberst, A., and Augustin, D. (2010). A model of aesthetic appreciation and aesthetic judgments. *British Journal of Psychology*, 95(4):489–508.

Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867.

Li Deng (2012). The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine*, 29(6):141–142.

Lim, B., Son, S., Kim, H., Nah, S., and Lee, K. M. (2016). Enhanced Deep Residual Networks for Single Image Super-Resolution.

Liu, D.-C. (2017). A Practical Guide to ReLU – TinyMind – Medium.

Lu, X., Lin, Z., Jin, H., Yang, J., and Wang, J. Z. (2014). Rapid: Rating Pictorial Aesthetics using Deep Learning. *Proceedings of the ACM International Conference on Multimedia - MM '14*, pages 457–466.

Luo, Y. and Tang, X. (2008). Photo and Video Quality Evaluation: Focusing on the Subject. pages 386–399. Springer, Berlin, Heidelberg.

Minitab (2017). A comparison of the Pearson and Spearman correlation methods.

Murray, N. and Gordo, A. A deep architecture for unified aesthetic prediction.

Murray, N., Marchesotti, L., and Perronnin, F. AVA: A Large-Scale Database for Aesthetic Visual Analysis.

Nam, H. and Han, B. (2016). Learning Multi-Domain Convolutional Neural Networks for Visual Tracking. Technical report.

Niu, Y. and Liu, F. (2012). What Makes a Professional Video? A Computational Aesthetics Approach. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(7):1037–1049.

Obrador, P., Schmidt-Hackenberg, L., and Oliver, N. (2010). The role of image composition in image aesthetics. In *Proceedings - International Conference on Image Processing, ICIP*, pages 3185–3188. IEEE.

O'Donovan, P., Agarwala, A., and Hertzmann, A. (2014). Collaborative filtering of color aesthetics. In *Proceedings of the Workshop on Computational Aesthetics - CAe '14*, pages 33–40, New York, New York, USA. ACM Press.

Owens, J. and Hunter, A. (2000). Application of the self-organising map to trajectory classification. In *Proceedings Third IEEE International Workshop on Visual Surveillance*, pages 77–83. IEEE Comput. Soc.

Panchenko, D. (2006). *Statistics for Applications — Mathematics — MIT OpenCourseWare*.

Park, K., Hong, S., Baek, M., and Han, B. (2017). Personalized image aesthetic quality assessment by joint regression & ranking. In *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, pages 1206–1214. IEEE.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks.

Raj, B. (2018). Data Augmentation — How to use Deep Learning when you have Limited Data — Part 2.

Rebuffi, S.-A., Bilen, H., and Vedaldi, A. (2017). Learning multiple visual domains with residual adapters.

Rebuffi, S.-A., Bilen, H., and Vedaldi, A. (2018). Efficient parametrization of multi-domain deep neural networks.

Ren, J., Shen, X., Lin, Z., Mech, R., and Foran, D. J. (2017). Personalized Image Aesthetics. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:638–647.

Rothe CVL, R., Zurich, E., Timofte CVL, R., and Van Gool Leuven, L. K. (2016). Some like it hot-visual guidance for preference prediction. Technical report.

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

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Sanchez, J., Perronnin, F., Mensink, T., and Verbeek, J. (2013). Image Classification with the Fisher Vector: Theory and Practice.

Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. (2007). Collaborative Filtering Recommender Systems. In *The Adaptive Web*, pages 291–324. Springer Berlin Heidelberg, Berlin, Heidelberg.

Schoch, E. (2001). *The Everything Digital Photography Book: How to Take Great Pictures*.

Sheppard, R. and Martin, B. (2009). Digital Photography Tips – National Geographic.

Shimoda, W. and Yanai, K. (2016). Distinct Class-Specific Saliency Maps for Weakly Supervised Semantic Segmentation. pages 218–234. Springer, Cham.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. Technical report.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. (2017). SmoothGrad: removing noise by adding noise. Technical report.

Sonoda, S. and Murata, N. (2015). Neural Network with Unbounded Activation Functions is Universal Approximator.

Srivastava, N., Hinton, G., Krizhevsky, A., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015a). Highway Networks.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015b). Training Very Deep Networks. *Neural Information Processing Systems*, pages 2377–2385.

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

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. (2016a). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going Deeper with Convolutions.

Szegedy, C., Vanhoucke, V., Ioffe, S., and Shlens, J. (2016b). Rethinking the Inception Architecture for Computer Vision.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. Technical report.

Tang, X., Luo, W., and Wang, X. (2013). Content-Based Photo Quality Assessment. *IEEE Transactions on Multimedia*, 15(8):1930–1943.

Targ, S., Almeida, D., and Enlitic, K. L. (2016). Workshop track-ICLR 2016 RESNET IN RESNET: GENERALIZING RESIDUAL ARCHITECTURES.

Torrey, L. and Shavlik, J. (2009). Transfer Learning. Technical report.

Varma, R. (2018). Picking Loss Functions - A comparison between MSE, Cross Entropy, and Hinge Loss.

Vogel, D. and S. Khan, S. (2012). Evaluating visual aesthetics in photographic portraiture. *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*, page 128.

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

Wang, W. and Shen, J. (2017). Deep Cropping via Attention Box Prediction and Aesthetics Assessment. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:2205–2213.

Wong, S. C., Gatt, A., Stamatescu, V., and McDonnell, M. D. (2016). Understanding Data Augmentation for Classification: When to Warp? In *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–6. IEEE.

Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K., and San Diego, U. (2017). Aggregated Residual Transformations for Deep Neural Networks.

Yan Ke, Xiaoou Tang, and Feng Jing. The Design of High-Level Features for Photo Quality Assessment. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR'06)*, volume 1, pages 419–426. IEEE.

Yu, F., Koltun, V., and Funkhouser, T. (2016). Dilated Residual Networks.

Yu, W. and Chen, X. (2018). Aesthetic-based Clothing Recommendation. 2.

Yunpeng, C., Xiaojie, J., Bingyi, K., Jiashi, F., and Shuicheng, Y. (2017). Sharing Residual Units Through Collective Tensor Factorization in Deep Neural Networks.

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method.

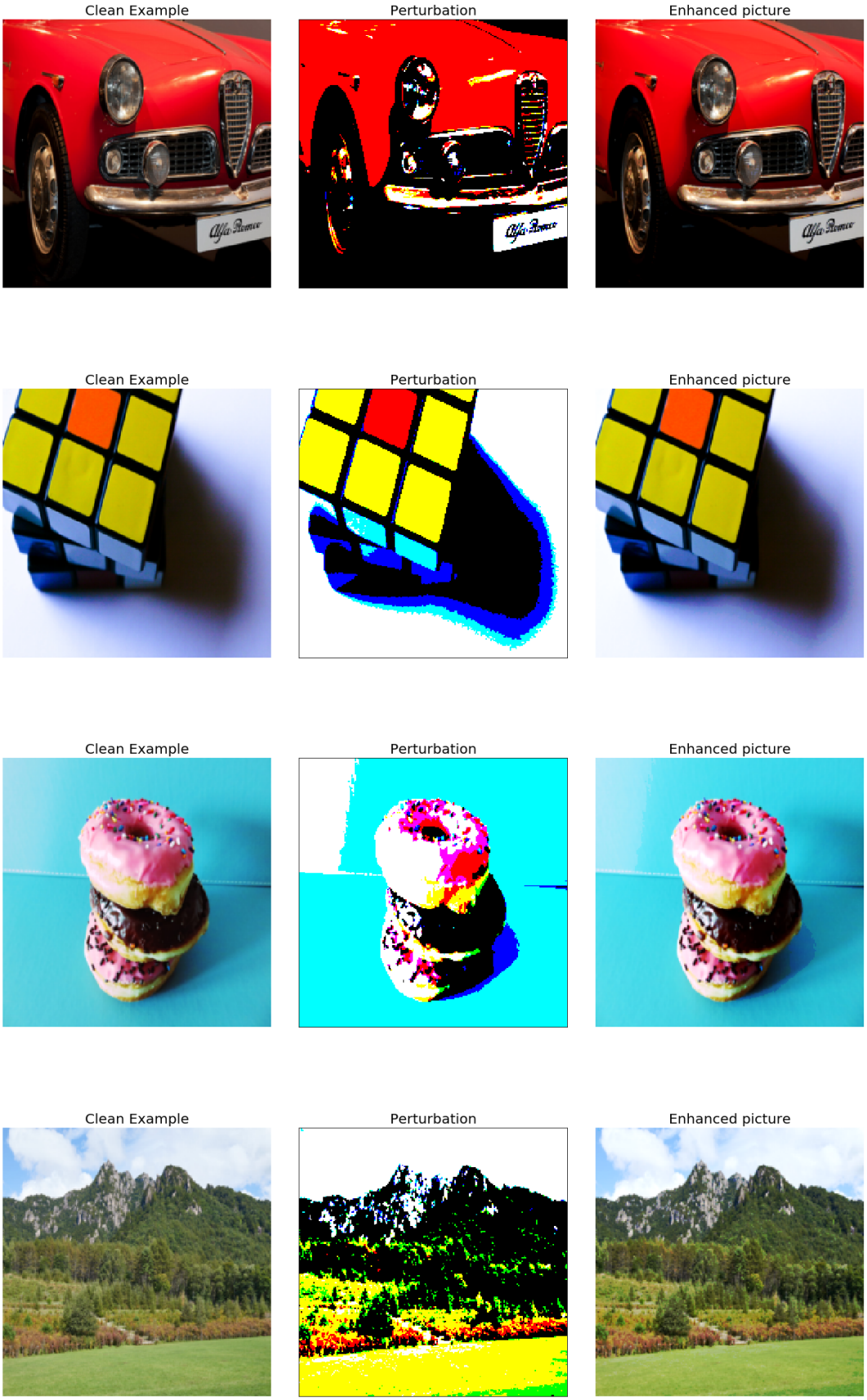
Zhang, J., Zheng, Y., and Qi, D. (2017). Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction *.

Zhang, K., Sun, M., Han, T. X., Yuan, X., Guo, L., and Liu, T. (2016). Residual Networks of Residual Networks: Multilevel Residual Networks.

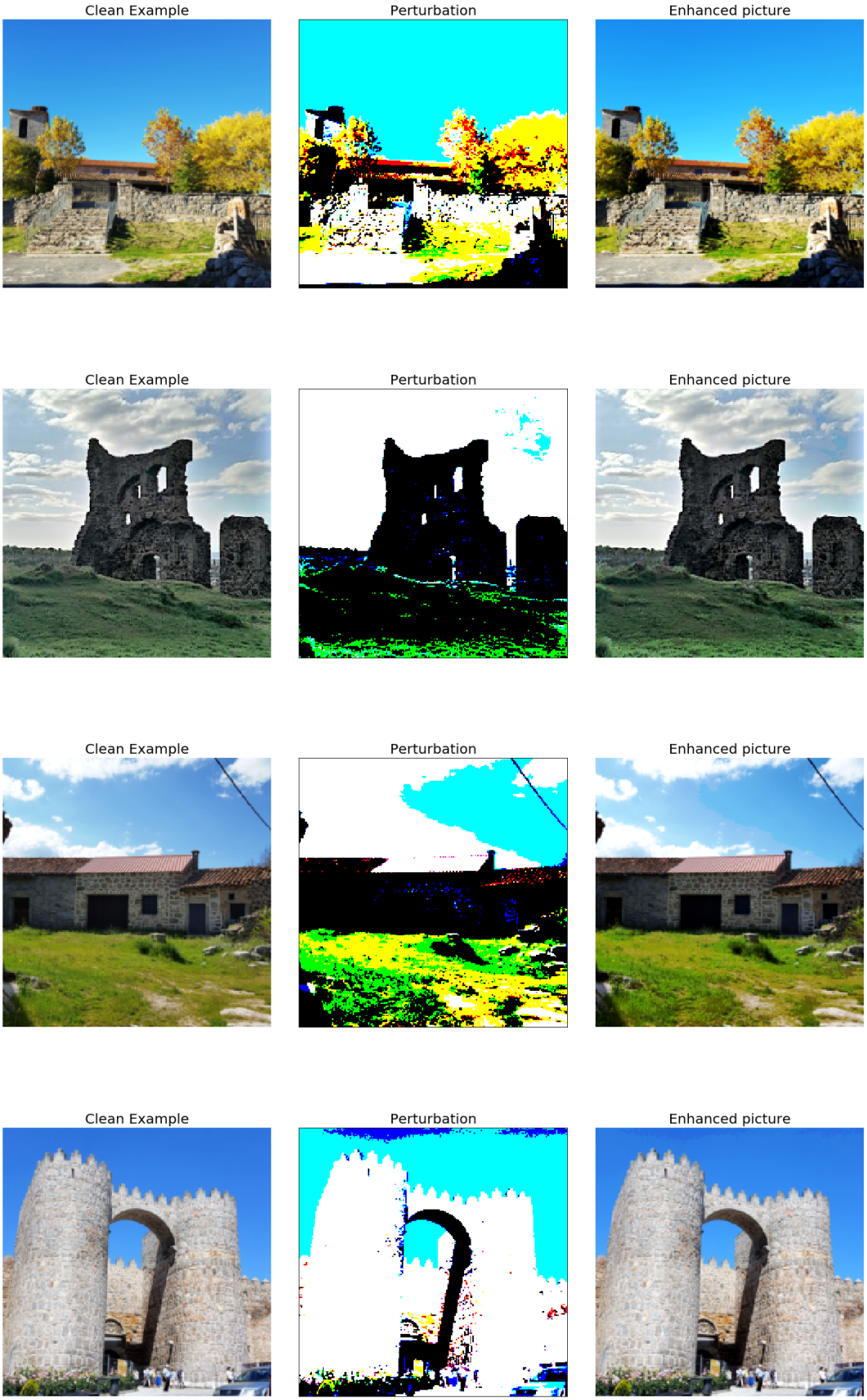
Appendix A

Additional figures

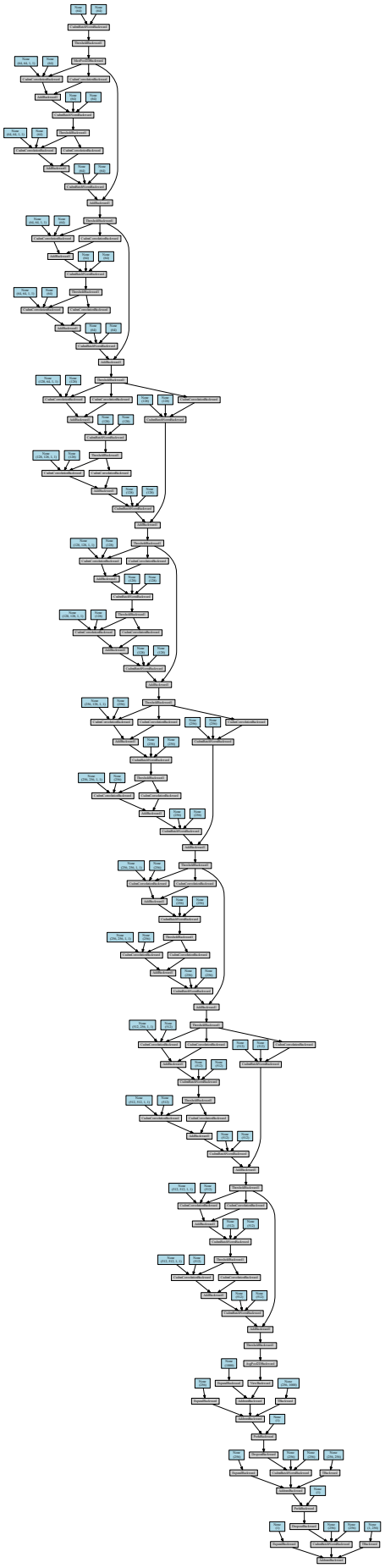
More examples of saliency maps using the mean aesthetic prediction network and pictures taken from the FLICKR-AES



Additional enhanced images using Fast Gradient Sign. The pictures were obtained from the Flickr-AES dataset. All the enhanced images had a greater expected rating (around 10% larger) than their corresponding original pictures.



Additional enhanced images using Fast Gradient Sign. The pictures were taken by the author of this thesis. All the enhanced images had a greater expected rating (around 10% larger) than their corresponding original pictures. This method has shown to work in pictures that did not come from the dataset used to train the network.



Autograd flow chart of the Resnet-18 network with residual adapters.

Appendix B

Colophon

This document was set in the Times Roman typeface using LaTeX and BibTex, composed with Overleaf. The bibliography was generated using Mendeley.