Fine-art Paintings Style Classification with Transfer Learning

University of
St Andrews

Yein Kim

Supervisors

Tom Kelsey, Carl Donovan

BSc (Hons) Computer Science and Mathematics

April 12, 2019

Abstract

Neural Networks are one of the most prominent types of design models in research in deep learning. Convolutional Neural Networks have been shown to be particularly powerful, excelling in image analysis. Application of CNN in tasks beyond object recognition has become a popular research area with the use of transfer learning: developing a new model by taking the existing model as a baseline. The new task is to investigate fine-art paintings classification in style. By designing and delivering empirical evaluation of models with different architecture, the project aims to analyse the transferability of features and the implication of learning methods on the generalisation ability of the new model. The resulting models provided promising results where transfer learning with fine-tuning have improved the model performance over a series of incremental retraining from the top in descending level of feature specificity. This learning approach helped in understanding layer-wise feature granularity in its contribution for an enhanced model. It is also found as part of the analysis that models encode different visual features as "generic" based on the architectural design from its base task of image recognition, which have influenced the model performance on specific classes for style classification where classes are characterised by various underlying features.

"I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated. The main text of this project report is 11,067 words long, including project specification and plan. In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the report to be made available on the Web, for this work to be used in research within the University of St Andrews, and for any software to be released on an open source basis. I retain the copyright in this work, and ownership of any resulting intellectual property."

Acknowledgements

I would like to express my sincere appreciation to the members of the School of Art History, who have provided generous support from data collection to contextual discussions, and who have contributed in the shaping of this project: Dr Natalia Sassu Suarez Ferri, Associate Lecturer, for advising on contextual analysis of the problem and stylistic relationships of the classes; Swithun Crowe, Research Software Engineer, for providing access to image data; Andrew Demetrius, Visual Resources Curator, for initial guidance at data enquiry; and Yejun Kim, second year Art History student, for advising on painting collation by theme and artist.

I would like to declare that all fine-art paintings included in this report and being used for this project are from WikiArt.org, an online fine-art collection in public domain and copyright protected as used for informational and educational purposes, and from the School of Art History Image Database in the case of any works which are not available in WikiArt.org.

I would like to pass on my sincere gratitude to my supervisors for their directive support and guidance throughout the duration of the project whenever needed.

Contents

List of Figures

List of Tables

Appendix: List of Figures

Chapter 1

Introduction

1.1 Motivations

Neural Networks are one of the most prominent types of design models in research in deep learning. Convolutional Neural Networks (CNN) have been shown to be particularly powerful, excelling in image analysis. The substantial groundbreaking work delivered in the field of visual object recognition in recent years via transfer learning has given rise to the question of its transferability to work with classification tasks of different nature.

The focus of this paper is to investigate its ability in capturing the underlying visual characteristics of fine-art paintings to determine its style.

With the ever increasing volume of digital art collections and wider accessibility of art works over centuries, building an highly accurate fine-art paintings classification model would lead to an effective automation to implement. Building a model with human-level equivalent accuracy is an ongoing challenge. It depends on how well the model mirror's a human decision making of style based on images.

1.2 Context of the Study

Initial studies in the research area involved combining visual feature representations and traditional machine learning algorithms. Recent studies on CNN have been successful in modelling more sophisticated classification tasks and provided state-of-the-art results related to style classification. A detailed survey of related work will be done in Section 2.3.

However, there is a lack of detail in training methodology including parameter configuration and analysis of the model performance according to architecture designs of the models. Justification behind its architecture design and training methods will be essential to build a model that can be widely used and is reproducible. Researches typically included an empirical evaluation with several architecture designs while further investigation would help understand how one architecture outperforms the other. The extent to which features can be shared between two different tasks are investigated.

1.3 Objectives and Contributions

The paper aims to implement transfer learning technique to build a CNN model specialised to fine-art paintings classification by style and compare the relative similarities in the features learned for object recognition and those for style identification of paintings.

After an interim revision of objectives by taking literature search into account, the objectives of the project are the following:

- A detailed literature search of the mathematical theory and use of CNN in image analysis and related work delivered in fine-art style classification. Identify key state-of-the-art methodologies to follow: transfer learning and fine-tuning.
- An organised and coherent empirical research report of image analysis from data preparation, algorithm implementation to result analysis.
- An analysis on different CNN architecture design and comparison of the relative generalisation ability.
- A heuristic model derived from experimentation with competent performance under common constraints in model building including space and time.

The secondary objectives are the following:

- Identify any limitations of the current designs and deliver artistic reasoning behind mis-classifications generated if any.
- Discuss a potential direction of improvement for an enhanced model by combining the result and literature.

1.4 Overview of the Project

Chapter 2 will describe the underlying mathematical theory of CNNs, including convolution operation and optimisation algorithms, a contextual definition of transfer learning and a thorough survey of research conducted to present on fine-art paintings style classification. Chapter 3 will project management for its delivery. Chapter 4 will discuss the methodology used for architecture selection, transfer learning and fine-tuning in various stages. Chapter 5 will deliver the experimental results, extensions and discussions. The final chapter will comment on the outcome from the perspective of a domain expert and conclude with future improvements.

Chapter 2

Context Survey

Summary

The mathematical theory of CNN will be explained, with a detailed survey of the method in building a high performing model from optimisation and regularisation techniques. This will be followed by the definition of *transfer learning* and the base task of visual object recognition. The final section will be devoted to related work in classification of fine art paintings as image data. It will discuss how the contribution of CNN and transfer learning have led to the state-of-the-art model capturing the artistic imagery beyond object recognition.

2.1 Convolutional Neural Networks

Artificial Neural Networks are one of the most widely used type of machine learning framework.[5] It involves a layer-wise learning, where linear combinations of outputs of one layer are inputted into the next: each layer consists of nodes with activation functions which will generate nonlinear functions as data proceeds. The output will be computed by concatenating all the functions from its penultimate layer, generating the optimal probability of prediction of classes/variables.[6]

The paper is particularly interested in the use and design of neural network for image classification, and factors contributing the learning process of such model in practice. These will be discussed in the following section.

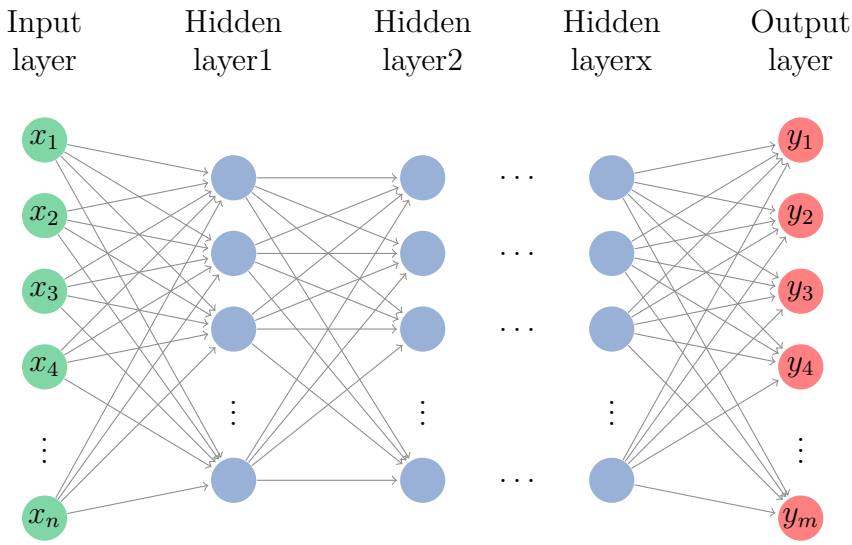

Figure 2.1: ***A Generic Sketch of a Feed-forward Neural Network [1]:*** *a deep neural network will involve a large number of hidden layers. For image analysis, the input layer will be a multi-dimensional array of pixel values, with hidden layers to include convolutional layers which would only be partially connected. For a classification problem, the output layer will involve n units for n classes for prediction where the preferred softmax function will output the probability of the prediction of input data belonging to a class i for an i^{th} node.*

Convolution

A CNN is a particular type of an artificial neural network which is shown to be a competent model for image processing, evident from its success as the forefront of image object recognition challenges of recent years.[7][8][9]

It includes a layer that exploits a convolution operation denoted as (*). As Goodfellow *et al.* derives, let $x(t)$ be a position function of the input and $w(t)$ a weighting function on the input, such that $w(t-a)$ will enforce values in the neighbourhood defined by a to make more contribution in the output of the convolution operation.[5] Hence, the output $s(t)$ is a weighted sum of previous positions in some range for a.

$$s(t) = (x * w)(t) = \int x(a)\, w(t-a)\, da \tag{2.1}$$

This original definition of convolution operation [5] is expanded by letting the

range to tend to infinity as in Eqn. 2.2.

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)\, w(t-a) \tag{2.2}$$

At a convolution layer, this is translated as a finite sum of product between the input data (I) and kernel (K), generating an output of the layer as a feature map (S), which are represented as multi-dimensional arrays.[5] Kernel (K) is a multi-dimensional array with initialised values to which input data is applied. The operation thereby reflects the relative influence of values in the neightbourhood of the input pixel position generating the product value. Hence, every entry of i^{th} row and j^{th} column of S ($S(i,j)$) will be the output of convolution operation applied on input $I(i,j)$.

$$S(i,j) = (K * I)(i,j) = \sum_{m} \sum_{n} I(i-m, j-n)\, K(i,j) \tag{2.3}$$

where m and n represent the length and height of a kernel, equivalently called a filter, respectively. It is a commutative operation thereby this reversed formula is more intuitive and widely implemented in libraries: it aligns with the illustration in Figure 2.2.[5]

Pooling layers are often implemented to follow convolutional layers in various CNN architecture designs.[5] Pooling is a downsampling operation such that pixel values located under the given filter window will be replaced by downsampled selection of values in the output.[6] The method reduces the dimensionality of feature maps, helping maintain low parameter size.[6] Stride (s) is used as a downsampling factor. Typically value of $s = 2$ with a filter size of 2×2 on feature maps of size $(N \times N)$ will output of size $(\frac{N}{2} \times \frac{N}{2})$: where values are chosen for every 2×2 pixel area of input feature map. The most standard pooling operation selects the maximal value to ensure no information is lost, called Max-pooling.[6]

Convolution operation allows to model non-linearities in the data by applying non-linear transformation, mapping the data to the new feature space, making linear separation possible.[10] Additionally, it has overcome a key limitation of feed-forward neural networks, which suffer from increased number of parameters in the fully-connected layers, deterring from the building of deeper models.[11] This is possible due to “sparse interactions” of units between each layer, where

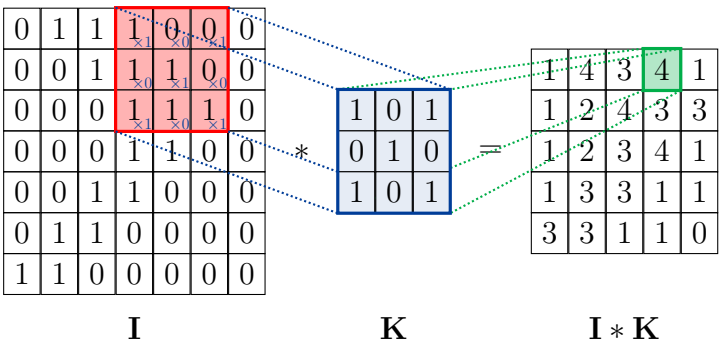

Figure 2.2: ***An Illustration of Convolution Operation[2]****:a visualised example of convolution operation defined in Equation (2.3). It is essentially a series of convolution between a region in the input, a receptive field, and the kernel; the kernel acts as a slide window in the input where the interval of translation of the window is defined by stride.[5] The size of the kernel and the value of the stride will determine the dimension of the output of convolution.[10]*

each unit of the following layer only interacts with the nodes from the previous layer within its receptive field.[5] Parameter sharing mechanism has also allowed a significant reduction in the number of parameters of the model.[10]

Optimisation

Training of the network is a variant of an unconstrained optimisation problem, where the objective function is an expectation computed over the unknown distribution of given data of the specified loss function as shown in 2.4. This is not a complete optimisation problem since the true underlying distribution of the data in the training set is unknown.[5]

As defined by Goodfellow *et al.*, for a given input value x and its true response y in a training set *data*, the objective function $J(\theta)$ with known parameters θ can be formulated as the expectation of loss computed over all examples provided with the computed distribution of data from training set as $\hat{p}_{data}$:

$$J(\theta) = \mathbb{E}_{(\mathbf{x},y)\sim\hat{p}_{data}} L\left(f\left(\mathbf{x}\,;\theta\,\right), y\right) \tag{2.4}$$

"where L is the per-example loss function", $f(x\,;\theta\,)$ is a predicted response given the parameters represented as θ and the input x, and $\hat{p}_{data}$ as the derived distribution of the data set.[5] The aim of the problem is "to minimise the expected

generalisation error", that is the empirical risk denoted as $J^*(\theta)$:[5]

$$J^*(\theta) = \mathbb{E}_{(\mathbf{x},y)\sim p_{data}}[\, L(\, f\,(\mathbf{x}\,;\theta\,), y\,)\,] = \frac{1}{m}\sum_{i=1}^{m} L\,(\, f(\mathbf{x}^{(i)};\,\theta\,), y^{(i)}\,) \tag{2.5}$$

where the expectation will be computed across the "data-generating distribution" $p(x, y)$ instead, given the sample of size m.[5]

The first order **Gradient-based optimisation** is used for training neural networks in this project. The gradient of the objective function is widely used to infer the direction - the first order derivative of $J(\theta)$. The location of a critical point is deduced by having a zero gradient at the location.[12] This property is exploited in the first order optimisation algorithms, which only compute the Jacobian matrix of first order partial derivatives of the loss function, with respect to the weights.[12] It is important that an appropriate choice of optimisation technique is used such that nonlinearity introduced by a neural network, resulting in a non-convex loss function, is dealt with appropriately in training.[5]

Backpropagation is used to compute the gradient of the objective function with respect to the parameters to be computed[5], such that weights can be updated in the opposite direction of the direction of the gradient, hence towards the minimum.[11] It is applied to parameter updates at the end of each iteration: changes are "backpropagated" according to the value of the loss function computed at the output layer.

The key optimisation algorithms to be experimented with are Stochastic Gradient Descent (SGD), *RMSProp* and *Adam*: as discussed in detail in Appendix A in terms of its design. Note there are other methods which are available in practice including the second order optimisation methods. Hence the background information of second order optimisation method is also included in Appendix A.

Loss Function

The loss function chosen is cross entropy (deviance) applicable to multi-class classification problem.[6]

$$L(\theta) = -\sum_{i=1}^{N}\sum_{k=1}^{K_i} y_{ik}\log f_k(x_i) \tag{2.6}$$

where the entropy is computed for every class of N classes per observation for up to K_i samples in each class, with y_{ik} as the true label of k^{th} observation in class i with $f_k(x_i)$ as the predicted response.[6] This is used to compute both training and validation loss using train and validation set respectively.

Regularisation

Regularisation is a holistic term for strategies designed to reduce the test error "with a possible expense of increased training error".[5] - to mitigate the effects of overfitting.[10] Overfitting is caused by failure to make an appropriate compromise between model complexity and generalisation ability for many learning models, where increased model complexity produces low training error while high testing error; this is due to the model becoming more capable of fitting not only the signal but noise.[10]

Analogous to the addition of penalty terms in the objective function for other statistical models such as linear regression, adding a norm penalty $\Omega(\theta)$ on the objective function is one way of applying regularisation.[5]:

$$J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) = J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) + \alpha\, \Omega(\boldsymbol{\theta}) \tag{2.7}$$

The value α determines the level of regularisation applied to the objective function with 0 denoting the non-regularised original objective function.[5]

L2 Regularisation imposes the norm penalty as $\Omega(\boldsymbol{\theta}) = ||w||^2$ such that the effect of the regularisation is a decay of the weight (w) by a constant multiplicative factor prior to the gradient update; this will make the weight terms for every unit to decay close to 0.

Dropout is another type of regularisation technique:it disables a proportion of units from learning completely from one iteration to the next on a layer.[10] The implication of this is an ensemble training of various subnetworks from the base network. This differs with the usual bagging technique, since all subnetworks share parameters such that the resulting network will be an amalgamation of learning of the terms over a series of iterations.[5]

Early Stopping is a type of regularisation method which can be used as part of training procedure. As the number of iterations increases, it is more likely that the model learns the underlying noise to overfit. The key indication

of overfitting would be when validation loss diverges with training loss and training accuracy maintains their monotonic behaviour. Hence, by tracking the minimal loss or maximal accuracy for validation set, model training can be configured to terminate if undesired behaviour is spotted for a consecutive number of iterations: epoch is defined as the full exposure of the model to the entire training set.[5] This term will be used in the place of iteration as appropriate.

2.2 Transfer Learning

The definition of transfer learning is a methodology of applying an existing pre-trained model, trained under a basic task, to a new task: which could be a specialisation from the basic task or a different problem in nature.[13] Pan.*et al.* provides a formal definition of transfer learning as follows:[14]

Definition 2.1. (Transfer Learning)[14] Given a source domain D_S and learning task T_S, a target domain D_T and learning task T_T, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in D_T using the knowledge in D_S and T_S where $D_S \neq D_T$, or $T_S \neq T_T$.

Transfer learning can be categorised to various types according to the relationship of "source" and "target" domain and tasks. The type of interest in this paper is ***Inductive Transfer Learning***, defined by Pan.*et al.* as a model with transformed knowledge of the source task such that the target task can be evaluated, given that the labelled source domain are readily available and the tasks known to be related.[14] The source and target domain are different, so as their task. By Definition 2.1, "source" domain $\mathcal{D}_S$ is ImageNet data set, with "source" learning task of *Object classification* and "target" domain $\mathcal{D}_T$ is fine-art paintings data set with "target" learning task of *art style classification*.

Yosinski*et al.* also states that such technique is readily applicable to target tasks where the size of target domain is smaller than the source domain; which is the case where paintings data (target domain) is substantially smaller than the original object image data (base domain).

The deep learning community has seen a rise in the use of transfer learning considering the successful results exhibited by using pretrained models. [15] The most widely used "source" models for image classification are those attained from

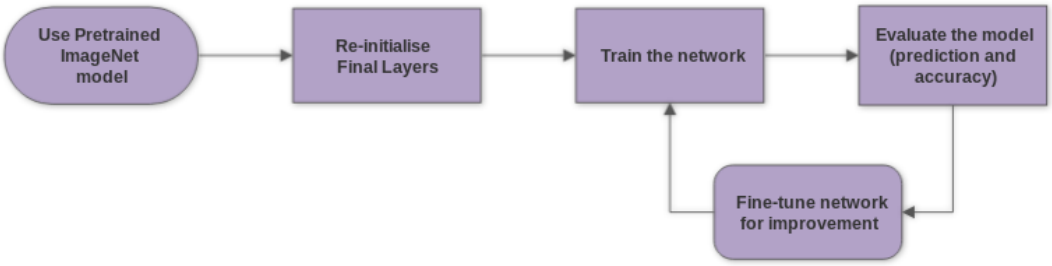

Figure 2.3: ***A Descriptive Flowchart of Transfer Learning Process[3]*** *: a typical step would include replacing a output layer with a new layer which is "specific" to the new task of interest and retrain the network to obtain a new model.[13] The subsequent retraining for improvement will constitute a fine-tuning process.[5]*

the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). It is an annual competition held for a number of visual recognition tasks whereby research teams participate with proposing architecture and algorithm, provided with the ImageNet data set.[7] The ImageNet data is a hand-labelled images of various objects, totalling over 14m images with over 20k object classes for classification. It has been used as a central metric for researchers to assess their proposed solution to attain the state-of-the-art results in tackling problems including object recognition, detection and localisation.[7]

Yosinski.*et al.*delivers a significant experiment to reason its success; the paper experiments with different configuration of models in order to quantify the level of generality of features for each layer and their translation from generic to specific within a network. 4 such models are trained to understand the implication of transfer learning and fine-tuning.[13] While transfer learning implied that all layers copied from the base network are kept frozen and only the new layers to be specific to the new task to be trained, fine-tuning involved retraining all layers. The number of layers of which weight are copied from in the base network are varied.[13]

The outcome showed, given that the two domains share common features transferable from one to the other, the retrained model performs well with incrementally improved performance with the increased number of transferred layers. However, this was only attainable with fine-tuning whereby the whole network

is retrained on the new model for smooth transition from the generic to high level features of the task domain;[13] the model with only replaced layers being trained exhibited degraded performance as more layers are copied from the base network. The explanation behind it is that the high level features of the source task have overshadowed the features required for the new task.[13] Additionally, Yosinski.*et al.* added that models trained with transfer learning outperform those with random initialisation of weights nonetheless.[13]

2.3 Related work on Fine-art Classification

Solving a new classification problem

Use of CNN as a feature extractor

Early studies into paintings classification can be exemplified by Shamir *et al.* who have used various types of Image Descriptors to identify useful visual features for painter recognition given paintings under 3 different styles.[16] These are then accompanied by CNN as one of many features extractors, similarly experimented by Bar *et al.*[17] Bar *et al.* extracted specialised features learned from CNN well trained on ImageNet data set,[7][8] utilising it as a generic feature descriptor to fuse with more low-level descriptor such as PiCoDes.[17] A similar investigation inspired by Bar *et al.* is conducted by Saleh *et al.* where features extractors are combined to create a new model for classification of style, genre and artist.[18]

Karayev *et al.* has extended the target domain from a data set merely containing fine-art paintings, to images in general including but not limited to, landscape and portrait photography in order to build a model which can recognise an image style in a more generic term.[19] The target labels encompassed atmospheric terms to describe images (e.g. sunny), colour (e.g. pastel), intensity (e.g. bright), to cinematic terms (e.g. noir, vintage,romantic).[19] The motivation behind the investigation is more focused on its potential practicality in visual search tools whereby the model's ability to capture similarities of various visual elements (colour, pattern, intensity) is significant.

The aforementioned investigations are common in the fact that CNN was yet to be accepted as a classification model per se, instead being recognised merely as a separate tool to extract pre-learnt knowledge.

Use of CNN as a classification model

Prevalence in the use of CNNs along with the help of transfer learning techniques led to other visual classification tasks.[15] The key interest lies in the extent of which the common features derived from object recognition tasks (ImageNet classification) would be applicable such that the new model could generalise to perform well with other types of classification tasks such as image/painting style recognition.[20][21]

Experimentation and Development

Tan *et al.* executed a stand-alone experiment with pre-trained CNN on ImageNet to build a model for painting classification in terms of style, genre and artist.[20] The work is based on transfer learning and compared the effect of stand-alone transfer learning and with conjunction to fine-tuning,[20] inspired by the experiment of Yosinski *et al.*[13]. The architecture used for CNN model is AlexNet, the winner of ILSVRC 2012 competition for ImageNet classification.[8] It is significant for being the first CNN model to participate in the competition.[7] The result of end-to-end training of the model (fine-tuning) and without fine-tuning have conformed in that transfer learning helps in better generalisation, with the former outperforming the latter.[20] The result has also coincided with the conclusion drawn by Yosinski *et al.* that the combination of transfer learning with fine-tuned parameters to the new target task have resulted in further improvement.[13][20]

Latest Experiment and State-of-the-art result

Lecoutre *et al.* further extends the investigation delivered by Tan *et al,* using a different type of CNN architecture that is Residual Neural Networks (ResNet), the winner of ILSVRC 2015 competition.[21][9] A more in-depth analytical approach to style classification is taken by investigating the optimal number of layers to retrain given the source model.[21] The key observation has agreed with the previous investigations, reinforcing the effectiveness of transfer learning[13] by additionally comparing models retrained from a pre-trained network to the model trained directly with paintings data set from scratch.[21] The latter has provided an empirical result to support the argument of Yosinski *et al.* that a model with weights optimised for object classification produced superior results

against a model consisting of randomly initialised weights, with what-ever degree of layers the weights are copied from.[13]: likewise supporting the argument that the two tasks are related since the features of the former has contributed to the target task. Overfitting is shown to be a major issue for building a reliable model which was the case for models built from scratch and in the case of retraining too many layers.[21]

Research in the past year included further analysis of the task domain. Despite of ongoing interest and effort in art style classification, there has been little prior analysis delivered by the community regarding how the new classification tasks differ to object recognition and what challenges to encounter. This is typically well addressed by Elgammal *et al.* which will be further explained in Section 4.1.[22] Elgammal *et al.* employed a similar transfer learning technique as Lecoutre *et al.*, while the key differences was that the baseline model used for transfer learning was the original model including the top dense layers encoding high level features for image object.[22] This enabled transferability can be explored as high as the domain specific features level. The result was yet again a state-of-the-art model suggesting that transferability is also applicable for high-level features as well. The outcome of comparing different architecture designs suggested that deep models did not win by high amount against shallower models implying that very deep models may not necessarily be the optimal model in the target task.[22] (less than 1% difference between optimal models for VGG16 and ResNet152 in accuracy)

The Table 2.1 summarises the performance of key models developed in the area with state-of-the-art results.

Table 2.1: *Summary of Style Classification Model Results Reported from Research*

Proposed	Type	Architecture	Method	Classes	Accuracy
Tan et al.[20]	CNN-AlexNet like	Sequential, replaced FC	pretrained only	27	46.0
			pretrained+ fine-tuned		54.5
Florea et al. (Pandora)[23]	Visual Features	SVM		12	54.7
		Random Forest			54.0
Lecoutre et al. (RASTA)[21]	ResNet50	50 Conv + replaced FC	pretrained only	25	49.4
			pretrained+ fine-tuned		60.1
			pretrained+ fine-tuned+ bagging		61.1
Elgammal et al. (2018)[22]	VGG16	13 Conv + 3 FC	from scratch	20	51.6
			pretrained+ finetuned		60.1
		additional 2 FC	from scratch		55.2
			pretrained+ finetuned		**62.8**
	ResNet152	152 Conv + 2 FC	pretrained+ finetuned		**63.7**

Chapter 3

Project Management

Although the project is not a software development project, it is managed using a number of tools for planning, progress tracking, and recording.

The project is divided to several stages with the lowest level being a sprint. The initial product backlog is created with each sprint running from 1-3 weeks with sprint objectives. These are all managed as *"Tasks To Do"* on the Trello board. The board was fully available to monitor for the supervisors. Regular supervisor meetings are used for checking progress on interim of a sprint or at the end with points of discussion being raised.

Version control is used to maintain the working environment with code, accessible both from mercurial managed by the School and via Bitbucket. The repository is directly connected with the Trello board: each task completed or to be done was referenced with the related commits made which helped in issue tracking.

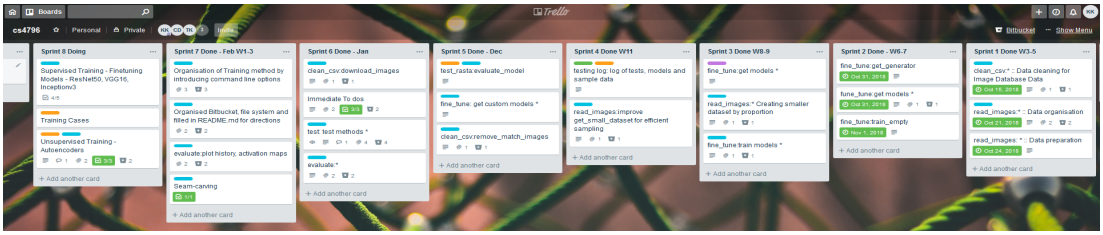

Figure 3.1: ***A Screenshot of Trello Board for the Project*** *: every sprint is categorised as a list consisting of tasks to do, each task is labelled as one of research, implementation and testing.*

The following road map shows an overall project journey which is a condensed

summary of the Trello board.

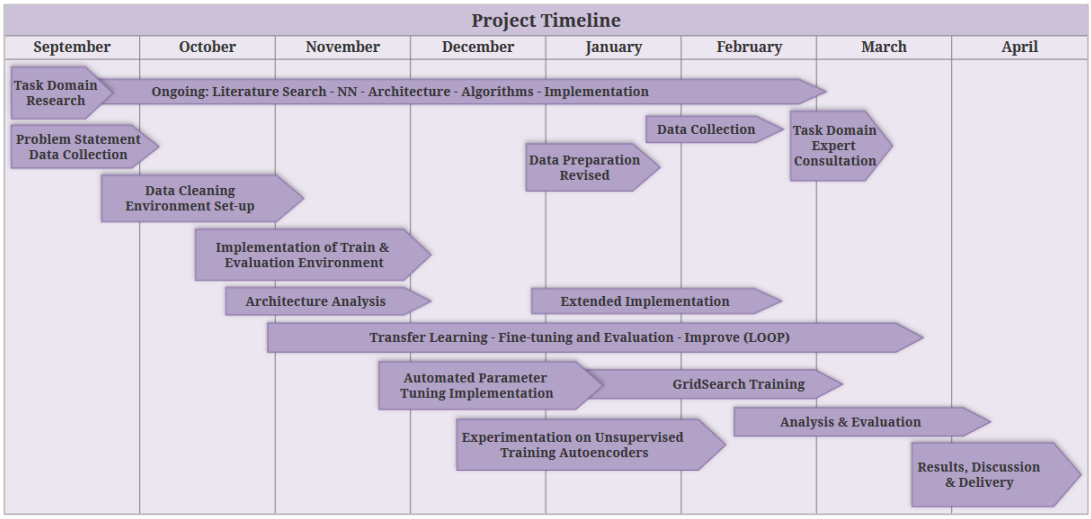

Figure 3.2: ***A Project Roadmap*** *: a general overview of the project timeline*

Chapter 4

Methodology

Summary

This chapter covers paintings data description and its class analysis, design and implementation of working environment used to conduct model training and evaluation. Transfer learning and training methodology will be outlined.

4.1 Dataset Descriptions and Analysis

The main source of data is wikiart.org[1], the largest online fine-art paintings collection available in the public domain and for educational use. Lecoutre *et al.* has provided the wikipaintings data set as part of the available source, which is used for training and validation in this project. This is to ensure that the invariance of the data set would allow more reliable comparison of models from their research and the resulting model produced in this project.

The full wikipaintings data set contains over 80k images from which 8:1:1 ratio of train:validation:test sets derived. Training set of the full data set contains 66,549 images of 25 classes. The styles encompasses from early 15^{th} century to the late 20^{th} century. The Figure 4.3 shows the counts of examples by class. It shows that the data set is clearly imbalanced, with one as large as the factor of 10 of the other. Data imbalance is common for real-life data and is also one of the factors to take into account in the later experimentation stage. The way in which imbalanced data set is handled will be discussed in the next chapter.

[1] https://www.wikiart.org/

It is assumed that all the proceeding results of the model performance will be subject to this specific arrangement of train, validation and test split and the choices of the 25 classes.

Analysis on the Definition of Style Classification and Challenges

The following points can be made regarding paintings data:

- Style classification is more challenging than object recognition since it involves predicting how a subject matter is depicted not what (subject matter) is depicted.[24] The Figure 4.1 gives some intuitive examples to illustrate this where the subject matter can be one of object recognition task label but according to the way it is portrayed, in terms of colour, intensity, tone, texture the class to which it will belong will vary. All these works may count towards the object class which is not the case for style classification.

Figure 4.1: ***Example paintings of common subject matter painted in different styles*** *: shows the key difference in object recognition and style classification (categorisation)[24]*

- Style classification have classes which are not fully mutually exclusive. The decision boundary of categorisation has varied with challenges through time

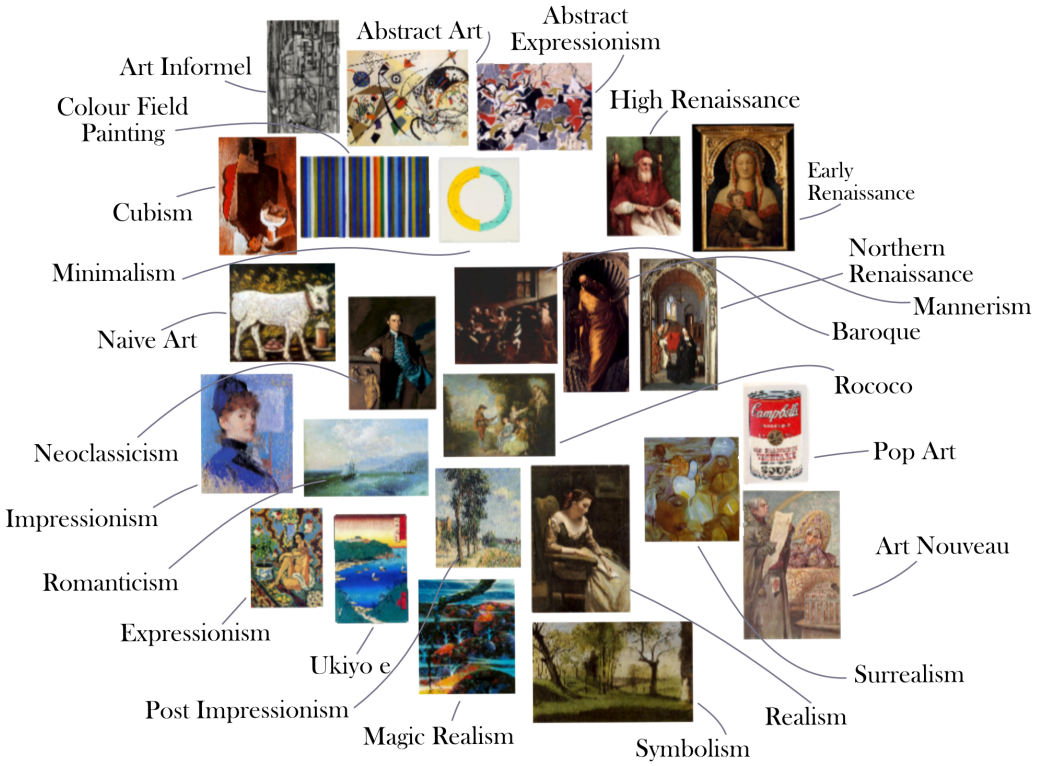

Figure 4.2: ***Example Images of Paintings tagged with 25 Style Labels from Wikipaintings_full Dataset*** *: they are grouped in the way in which classes are related in terms of time period, visual aesthetics and contextual connotations.*

by art historians, hence, it is still under the pool for extensive research in terms of capturing such factors in a learning model which requires specific domain expertise.[22] [24] Hence, the relationship of one style to the other can be as close to the point where one style defines the time period of transition from a different style to another, such that it shares the qualities of both. If one has stemmed from the other, they are likely to share subtle visual characteristics. Hence, the similarities may derive from the time period, artists working with multiple artistic styles, and values of the time such as "religious, social, moral and political" factors.[25]

– Pertaining to the classes for this data set, *Abstract Art* can be a set including a number of sub-classes and itself: Abstract Art, Abstract Expressionism, Art Informel, Colour Field Painting, Minimalism. 5

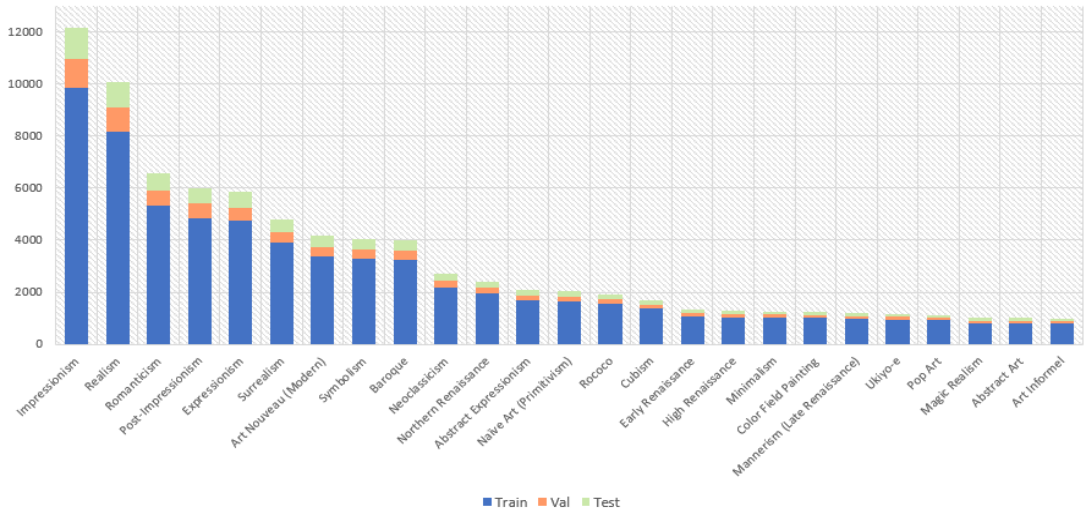

Figure 4.3: ***Number of Examples per Class grouped by Set Types for Wikipaintings_full Dataset*** *: the bar chart shows the number of data points per class of 25 classes in style from Wikipaintings_full data set. It is an imbalanced dataset.*

components are closely related in terms of its transcending values of non-figurative and geometric visuals. It is also comprised with sub-categories such as Abstract paintings in *Expressionism* and *Surrealism*. Cubism is also considered to be interwined. The key caveat in learning discriminative features would be having a closely related sub-classes with substantially smaller number of examples compared to those which are less related but with larger data sets.

- Another factor to incorporate is the diversity of artists for each class. Not one class is comprised with a single artist and it becomes rare from the period of Modern Art that that a single artist is attributed to the development of a single art movement which affected their style. This can be seen from the Figure A.4.

  As many artists have shown artistic development through time, works of art we see as their legacy embody subtleties in their visual characteristics for styles they worked in. Hence, it is important that the key characteristics of each style is not overshadowed by the individuality of the artists. This conforms with the early definition of style defined by Carl Friedrich von Rumohr in the late 18^{th} century which partly contributed in the evolved

definition of style; Rumohr's use of style was "a designation for an artist's individuality that transcends itself and becomes universal".[26]

Ultimately, the line of classification is even more challenging: the renowned art historian of 20^{th} century, Meyer Schapiro (1904-1996), stated artistic style as a revolving "constant form, elements, qualities and expressions in the art of an individual or a group."[25] This is the factor which the model should capture.

Elgammal *et al.* (2018) provided a good description on the challenges in fine-art style classification in the art domain's perspective.

– There is a large scope of information loss in the process of training a neural network model such as image pre-processing. This is due to the model requiring an image of fixed, hence any input image should be rescaled, introducing geometric distortions affecting its texture qualities.[22] Additionally, by setting all the works to a fixed size, any deductions which could be made by the original size and ratio of a painting in determination of its style is also lost.[22]

Additional Dataset

With the generous help from the School of Art History, an independent paintings data set has been collected from the School's online image database, which contains more than 60k collection of images spanning from classical paintings to installation. The condensed data set of paintings used for the project is comprised of over 30k images of unlabelled data set of works with wider variety of styles. The data set has been used for experimentation of unsupervised learning and prediction on trained models: attempts made are explained further in Section A.

4.2 Working Environment

The project environment is configured with tensorflow framework with keras in python for neural network modelling and training. The use of GPU is supported for training large number of parameters with training data. Model training is conducted at GPU-supported Scientific Linux lab clients at the School with GeForce GTX 1050 6GB and initial experiments are conducted in GeForce MX 150 2GB of a laptop.

The key dependencies are the following packages:

- **Tensorflow**[1] A main source package for training machine learning models (tensorflow-gpu)
- **Keras** An open source library for building neural network models The model architecture is pre-built, hence, time-efficient and flexible to extend the design. This is to ensure the tests are convincingly reproducible and environment is applicable to all users.
- **Keras-applications** A package required for accessing pretrained models; to instantiate models with ImageNet weights for transfer learning
- **Tensorboard** A library required for training monitoring and visualisation
- **Scikit-learn** A major python machine learning library required for preprocessing

The aim of the experimental approach is applying transfer learning with fine-tuning, by building an intuitive working environment for model training and evaluation using a large data set. The framework used for this project is largely inspired from those used by Lecoutre *et al.* in RASTA.[21]

4.3 Design and Implementation

The key design decisions are made throughout the project duration as each stage has required different series of implementation to facilitate the experiments.

Data Preparation

Data pre-processing methods are implemented to facilitate the training.

(a) **Data Pre-processing steps** for the *wikipaintings* data feed into the *ImageDataGenerator* when it comes to training. The major operations applied are mean image subtraction and image scaling. All 3 architecture are uniform in input shape of (224, 224, 3) - the default in the implementation.[9][27][4]

The above uniform pre-processing operations are set in order to conform with those from the existing ImageNet trained models. Hence, the mean from the original ImageNet training data set is used.[7]

[1] https://www.tensorflow.org/

Additional options for horizontal flipping and shifting are supported to include distortion if required for data augmentation, while the scaling has included distortions which was considered as a sufficient factor; this was the case since the resolutions of images vary greatly with substantial amount of works having an extreme width:height ratio. These options have been derived from the base environment of RASTA. Additional functions are written to facilitate the process, namely outputting data summary including counts, artist distribution across styles, and data sampling for k-fold cross-validation.

(b) The **Image Database**, (*ID*), data was collected as a csv file consisting of image data entries, metadata, followed by access urls for downloading images in different resolution types. Every image entry has been included manually without a concrete naming conventions and categories, hence, the data required substantial amount of cleaning. *processing_clean_csv.py* is devoted solely to functions for data cleaning and organisation. The key steps included: character replacement, removing redundant entries, allocating uniform naming conventions for data, artist name and title.

(c) **Model preparation** functions from *train_utils.py* are used in *train.py* for training. Functions include pretrained model initialisation, new model initialisation according to input configuration, and optimiser initialisation. The pretrained models are directly sourced from Keras-applications. It is designed with maximum flexibility to allow addition of dropout, regularisation and new optimisation types.

RASTA - Base Environment

The base environment from RASTA[21][1] supports training of models with command line options to define basic parameters including the number of epochs, batch size and drop out rate if any. In this project, the environment is extended such that more of a variety of hyperparameters can be controlled from command line as tuning parameters instead of built in configuration at training execution. The section describes the functionality provided by RASTA before introducing how it is extended for this project.

[1] https://github.com/bnegreve/rasta

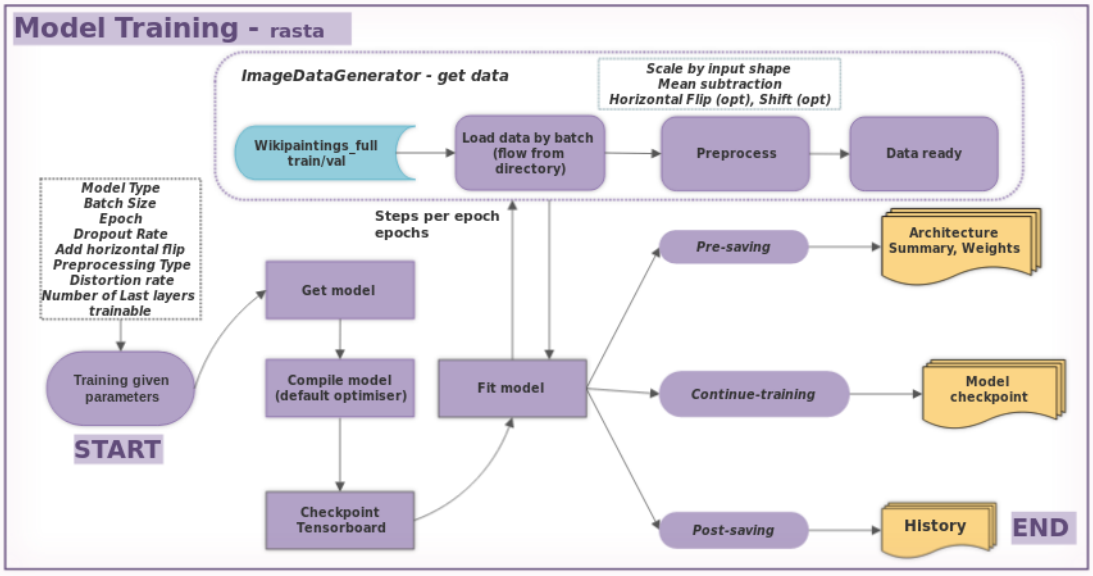

Figure 4.4: ***A Flowchart for RASTA Train Implementation*** *: describes the train process for RASTA*

1. Initialise the model according to the choice of architecture. It supports architecture types of: AlexNet, ResNet, custom built ResNet-like architecture. An option of adding dropout inside the last residual block is provided.
2. Set the specified number of layers trainable and compile with a default optimiser: *RMSProp* with a learning rate of 0.001.
3. Pre-save information about the model.
4. Train the model from scratch or with a given model path, continue training previously saved model from the saved state of the optimiser.
5. Post-save the model including model training history and the final model.

In the process of preparing model fitting, some key recording tools of model checkpoint and tensorboard adaptation are added.

As mentioned from [21], Lecoutre *et al.* have used Keras package for major operations such as *ImageDataGenerator* for data generator and *fit_generator* for fitting. Considering the volume of image data set handled, data is loaded in memory by batches denoted as a step, with every iteration (epoch) consisting of $\frac{sizeoftrainset}{batchsize}$ steps.

– *Model checkpoint* : It saves the state of the model including the trained weights at the end of every iteration. This helps in selecting the optimal state in the case of overfitting. The metric monitored for the checkpoint to occur is validation loss and validation accuracy.

– *Tensorboard* : Tensorboard allows to monitor model training using its graphical representation of loss function. The trend in training and validation loss and accuracy can be tracked acting as a comprehensible guide to understand the behaviour from early on.

Additionally, evaluation methods are available such as making predictions on a single image or on a test set to find top k accuracy.

Project Environment

This is the environment that is used for all model training for this project. The implementation is inspired by and extended from the base environment.

Training

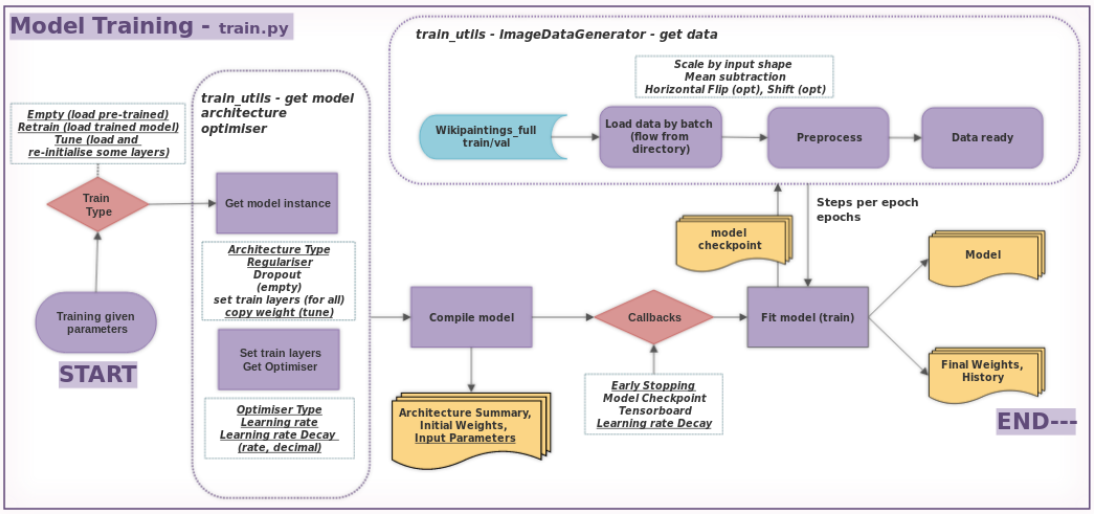

Figure 4.5: ***A Flowchart for Train Implementation*** *: describes the training process including interaction with model and data initialisation processes*

1. The type of training are defined as follows:
   (a) **Empty** (default): Train a new model of a specified *model type*, with pretrained or randomly initialised weights and optional figuration of *dropout* rate and addition of *regularisation*.

(b) **Retrain**: Retrain a model from the saved state of the optimiser. The model will only be recompiled with a new optimiser if
   - the number of trainable layers is changed.
   - the type of optimiser or the value of learning rate is changed.

(c) **Tune** : Train a newly initialised model with *specified layers copied* from a previously trained model, with *new trainable layers*. This combines the previous two options whereby model configuration and optimiser initialisation parameter options can also be passed in the command.

2. Along with extended model architecture configuration options, the model can be trained on a different choice of *optimisers* and initial *learning rate* to explore the convergence power of each.
3. Additional callback options are added including:
   - *Learning rate decay* can be scheduled for 2 types: step decay and exponential decay. This is used for training with SGD optimiser.
   - *Early Stopping* is added as a mean of monitoring and regularisation purposes. Validation loss of every iteration (epoch) is used to monitor with a default value of n = 3 to 5 iterations after which no change of minimal loss would lead to training termination. The value of n has also changed having learnt the initial model behaviour.
4. The final model is saved and can be analysed using *evaluate_result.py* with following options:
   - Predict on the test set to generate top k accuracy scores (from RASTA).
   - Predict on a single image (from RASTA)
   - Evaluate the model using the performance metrics: *confusion matrix*. The confusion matrix is suitable for evaluating the model generalisation for individual classes[6], whereby key areas of misclassification can be identified easily and reflected on the subsequent training.
   - Further evaluation tool such as interactive plots of accuracy and loss, precision and recall curves.

  - Further visualisation tool such as *plotting activation maps of convolutional layers* for a given model is implemented. It is typically useful for understanding features which filters of convolutional layers capture.

More details of evaluation tools will be described further in Chapter 5.

Automated GridSearch Training

Since the project takes an empirical approach using hyper-parameter tuning, it is considered necessary to build an environment with maximum automation. One approach taken for hyper-parameter tuning for many machine learning algorithms is GridSearch. It is an exhaustive search of given combinations of parameters to find the values producing a model with optimal result. However, this naive search is inefficient for tuning neural networks due to greater time and space complexity in comparison to other machine learning algorithms; additionally the amount of data used for training also complicates the matter.

Nonetheless, it can provide a good starting point to understand the trends with shorter iterations given a large number of parameters to compare: then it would be adequate in identifying patterns and verifying the implications of changing values of one parameter to the other. Hence, the following design of GridSearch is proposed and implemented in this project to assist in the empirical evaluation.

The following Figure 4.7 summarises the flow of GridSearch Training and Cross Validation (minor implementation) implemented in *train_hyp*. The key design decisions based on the requirements are addressed:

- Considering the size of data handled in this project and that the training is done by loading batches of images for every step, the general GridSearch implementation from machine learning library such as Scikit-learn was inadequate; it is written under the assumption that all data can be fit at once. Hence, this implementation is designed to support batch learning of model under GridSearch.
- It is designed to be fully supporting the functionality of GridSearch of packages. Hence, given the parameters and the values to be searched, it should output comprehensible result according to the value monitored to determine the optimum.

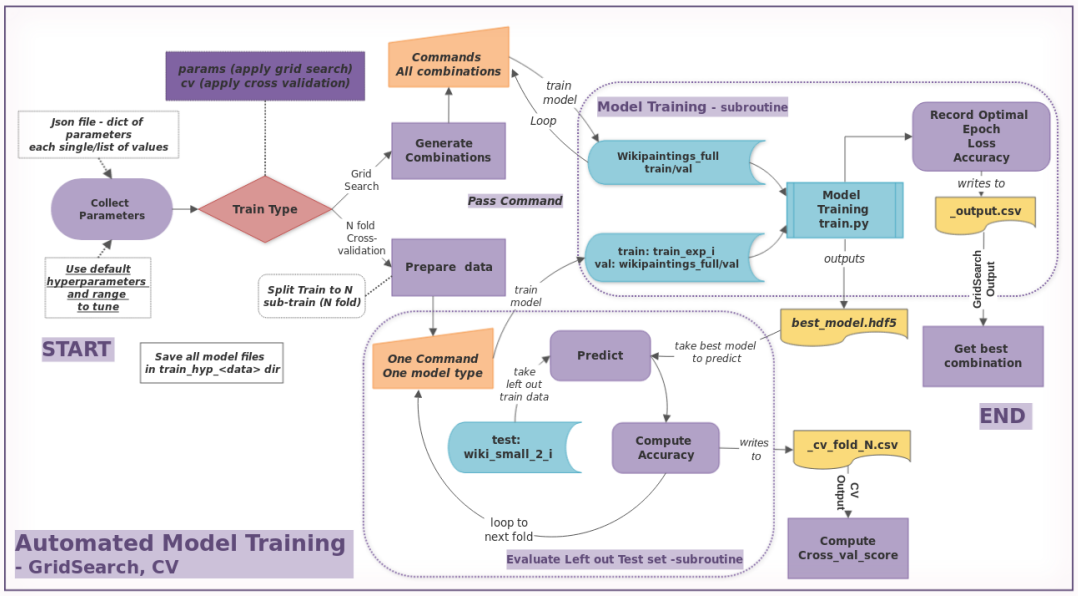

Figure 4.6: ***A Flowchart for Automated Train Implementation*** *: describes the initialisation and train process including interaction with model and data initialisation processes*

- For the ease of use, user defined parameters and list of values can be added as a JSON file which the script *train_hyp.py* accepts. When the file is not given, it automatically searches through the default dictionary, written with starting parameters to train a model with.
- By generating all the combinations of parameters, it uses *train.py* as a subroutine for training. The outcome of training will be generated as a results file (a standard output or saved in an output JSON file if *-s* save flag is set), reporting Top-3 combinations for optimal result. The performance criteria is set to be validation accuracy by default.

Additional implementation is K-Fold Cross-validation using another option *-t cv* and *-cv K* given one combination of model parameters to build the desired model. Similar to the execution flow of GridSearch, command is sent to *train* to be run as a subroutine. The prior preparation involves creating temporary directory for train and test split of the original training set. After every fold, the output is then again written to csv file to be analysed at the end with a cross validation score.

4.4 Methodology

Choice of CNN Architecture

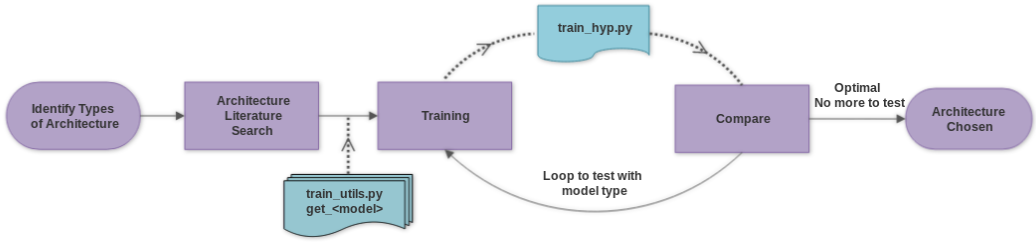

Figure 4.7: ***A Flowchart for Architecture Selection Process*** *: train a number of CNN models with different architecture to find the optimal.*

The Figure 4.7 describes the architecture selection process. Amongst the participants of ImageNet competition of past years, CNN models which have won the competitions are considered[7]: VGG, GoogLeNet and ResNet.[27][4][9]

The models have shown highly competent performance in the () 2014 and 2015 respectively.[7] Since AlexNet has set the bar high up with the strength of CNNs in 2012 by bringing the top-1 error rate in object classification task down to 16.4%,[8] the subsequent years have seen a majority of models for the competitions to be CNN - from which these architecture types are proposed. According to the Figure A.2, it is prominent that the later models largely contributed to the further reduction of error rate, reaching the state-of-the-art models in object recognition.

For this project, each architecture was assessed by their design, efficiency (memory, convergence speed) and tuning efficiency where contextual information are fully researched and summarised in Appendix A. The model is selected by considering both theoretical results and practical experimentation. The implemented models from Keras are VGG16, InceptionV3, ResNet50.

Transfer Learning and Experimental Fine-tuning

The process is built in reference to the Figure 2.3.

Stage 1: Obtain a pretrained model.

Stage 2: Re-initialise final layers - the layers are introduced for experimenting on the use of dropout and regularisation methods in the top layers. The initial design is modified after the training results.

Stage 3: Train the network using GridSearch and fully with more iterations. GridSearch involves training with different combinations of parameters with a small epoch size of 5. The output of the parameter training is then analysed to identify the combination which has given

- a converging training accuracy with reliable validation accuracy
- no sign of divergence for training and validation loss
- no sign of, or a moderate behaviour of, if any, overfitting.
- a suitable memory footprint for training.

Stage 4: Using the chosen hyperparameters, retrain the model with increased number of epochs for convergence. Model evaluation in relation to chosen parameter values can be done at this stage and revise the model design. Both quantitative and qualitative analysis helped to improve.

Stage 5: Based on the previous results, the number of trained layers is increased by fine-tuning. The last step aims to understand the level-wise transferability of features specific to object recognition to those for art style classification and thereby improve the model to the new task, as inspired by Yosinski *et al.* and Lecoutre *et al.*. This will help address the question of how low-level, mid-level and high-level features would generalise.

Chapter 5

Experimental Results

Summary

Empirical evaluation has taken several stages from architecture selection, initial training on pretrained models, fine-tuning revision to the final model evaluation.

In the process of building such model the biggest challenges to combat were overfitting and degradation of test accuracy due to the imbalanced dataset. The consecutive analysis has shown that the possible solution to address the challenges is: a series of retraining of intervals of trainable layers in replacement to a single training with large number of iterations. The replaced training method gives a promising result, with the help of EarlyStopping regularisation technique. Other regularisation techniques have shown to penalise too much that underfitting has occurred. The experimentation also showed that each architecture should follow its original design to maintain its optimisation performance.

Generalisation error can be improved by employing learn and refine technique which has appeared as a reliable method with the latter refinement on classification made using class weighting scheme. Hence, two models for each architecture are trained based on this strategy, resulting in satisfactory generalisation results with potential to improve further to refine its misclassification rate.

5.1 Choice of Architecture

Considering the limited memory allowed on the resources available, the versions of each architecture with the smallest memory footprint are used. These

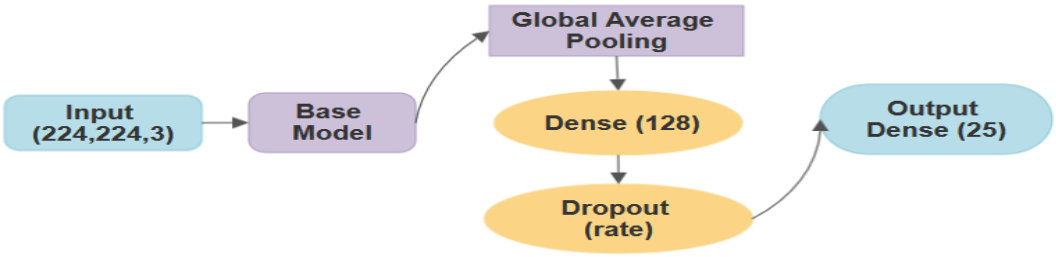

Figure 5.1: ***Initial Model Design for Architecture Selection*** *: Excluding the top layers of ImageNet trained models, all weights are initialised using ImageNet weights. Only newly added layers are trained.*

are VGG16, InceptionV3 and ResNet50 respectively. Models are built from the baseline pretrained models in Keras, where its top layers are replaced with a new series of Global Average Pooling layer, followed by 2 Densely connected layers, where the last layer is the Output layer with 25 output nodes for 25 classes.

Global Average Pooling layer which is not present in the original architecture allows aggregation of nodes from the previous layer onto a single dimensional vector.[28] Average pooling operation takes the average of all values within a receptive field. It is used over max-pooling operation to reflect the general representation of high level features of the model in the hope for better generalisation of the resulting model. Additional Dense layer as a penultimate layer helped in experimenting with regularisation techniques such as Dropout. Note that originally Dropout is not applicable to ResNet50 based on the original architectural design due to the use of intermediate Batch Normalisation layers which suffice regularisation: hence, this inclusion is used as an experimental point to explore.[9]

Parameters values are kept constant to compare solely with the difference in architecture.(Table 5.1 (L)) Prior contextual analysis helped in the choice of optimiser to be based on adaptive rate, with only last newly initialised layers to be trained to work with pretrained Imagenet models. Considering Lecoutre *et al.*' comment on overfitting and that this has been conducted for initial training, epochs were kept low with learning rate to be small to only accept small parameter updates.

The training of every epoch has taken approximately 45-60 minutes on average due to the size of training set, trainable parameters and the limitation in the batch

Table 5.1: *(L) Summary of Set Parameters, (R) Summary of Loss and Accuracy by Architecture Trained with 5 Iterations*

Parameter	Value
Epoch	6
Batch size	64
Layers trained	3
Learning rate	0.001
Optimiser	Adam
Dropout Rate	0.3

Architecture	Train Loss	Acc	Val Loss	Acc
VGG16	2.310	0.294	**2.150**	**0.333**
InceptionV3	2.662	0.204	2.801	0.179
ResNet50	2.307	0.295	2.315	0.303

size allowed under memory limitation.

The Table 5.1 (R) shows that VGG16 and ResNet50 have similar model behaviour up to the trained epoch while InceptionV3 significantly underperforms with high generalisation error. The relative underperformance of InceptionV3 model against other architecture types has been shown in subsequent experimentation where different parameter values are chosen within the search scope set for the project. This result implied that the efficiency derived from its base domain specific design [4][29] may not relatively optimise well in comparison to the optimisation ability of the others, as its learning performance was almost half of those from VGG16 and ResNet50.

5.2 Transfer Learning

Parameter Tuning

The key parameters considered are types of optimisation algorithm, learning rate, addition of dropout, decay and regularisation. The choice of learning rate value and the type of optimisation algorithm are prioritised since it is directly related to convergence and the speed of convergence of the loss function towards the optimal result.[11]

Table 5.2: *Summary of Parameters Tested in GridSearch : Stage 1*

Parameter	Learning rate	Optimiser	Dropout rate
Values Chosen	[0.001,0.01,0.1]	[Adam, RMSProp]	[0.3, 0.4, 0.5]

VGG16

VGG16 model is trained with a memory constraint thereby a batch size of 30 was chosen at the time of experimentation. In order to investigate initial training performance to select appropriate parameter values, epoch of 5 is chosen. Top-3 newly initialised layers are trained.

The Figure A.8 plots the training and validation accuracy of each model trained with different configurations annotated accordingly. The results show that:

- Addition of momentum variable in Adam optimisation method does not contribute largely towards its optimisation power; RMSProp and Adam perform equally well and showed correlated performance.
- Better generalisation occurs with smaller learning rate, there may only be small adjustments required to the random weights to interact well with the ImageNet weights of deeper layers.
- Dropout added in the penultimate Dense layer is most effective with the dropout rate of 0.3. It has a negative correlation against training accuracy.
- The second model with only the last Dense layer for classification output is included without dropout regularisation, has shown much better performance in terms of its learning speed. This suggests that this extra layer may not be necessary.

Table 5.3: *Summary of Top 5 Combinations with High Validation Accuracy for VGG16 after 5 Iterations: Stage 1*

	Optimiser	Learning rate	Dropout rate	Train Acc	Val Acc
1	Adam	0.001	0	**0.427**	**0.408**
2	Adam	0.001	0.3	0.362	0.355
3	RMSProp	0.001	0.3	0.364	0.341
4	Adam	0.001	0.4	0.315	0.331
5	RMSProp	0.001	0.4	0.307	0.326

ResNet50

Similarly, ResNet50 model is initialised with the model structure as shown in Figure 5.1 and trained in the same condition as for VGG16. In order to validate

the supporting argument of the exclusion of Dropout in the residual architecture, additional model is trained where Penultimate Dense layer is removed with no Dropout implemented.

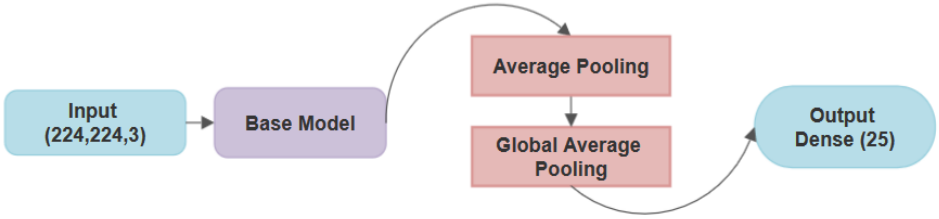

Figure 5.2: ***Revised Model Design using ResNet*** *: Excluding the top layers of ImageNet trained models, all weights are initialised using ImageNet weights. Only newly added layers are trained.*

As advised from the model design of Lecoutre *et al.* additional Average Pooling layer is introduced before Global Average Pooling layer in the removal of Dense layer, show in Figure 5.2.[21]. The outcome summarised in Table 5.4 suggests that while additional Dense layer with dropout mitigates overfitting, it is slow in convergence. There is no clear trend with varying learning rate as with VGG16 implying that further exploration is required. On the other hand, the second model converges well and quickly for training while the discrepancy suggests that overfitting is more apparent. The result from this experiment suggests that dropout regularisation technique introduces further parameters in its implementation and interferes with model training performance, hence, another type of regularisation technique would be required.

Table 5.4: *Summary of Top 5 Combinations with High Validation Accuracy for ResNet50 after 5 Iterations: Stage 1*

	Optimiser	Learning rate	Dropout rate	Train Acc	Val Acc
1	Adam	0.001	0	**0.564**	**0.402**
2	Adam	0.001	0.3	0.304	0.328
3	Adam	0.1	0.3	0.311	0.296
4	RMSProp	0.1	0.3	0.260	0.307
5	Adam	0.1	0.4	0.235	0.277

Comparisons with SGD and Adadelta optimisation methods are also conducted, as shown in Figure A.9:

Underperformance - AdaDelta AdaDelta is another type of optimisation algorithm that mitigates the issue of reduction in learning rate of AdaGrad differently to the design of RMSProp. It does not accumulate previous squared gradients to circumvent the issue.[5] However, the results suggest that the accumulation of previous average gradients do contribute in the learning towards convergence to the certain extent. (RMSProp and Adam have shown promising results).

Outperformance with Overfitting - SGD outperforms RMSProp and Adam in terms of its optimisation speed while this leads to faster overfitting. Overfitting is mitigated with largest dropout rate of 0.5. This can also be explained from the initial ResNet training methodology.[9] It is trained with SGD with high momentum and low learning rate. This suggests that despite of having the optimal learning performance within the time limit it is more likely to overfit.

Additional experimentation explored the influence of increasing batch size with different combinations of optimisation algorithm and learning rate. For invariant comparison, epoch is set to 5 with dropout rate of 0.3 using SGD optimisation algorithm: despite its overfitting behaviour it also wins in attaining highest accuracy with decreasing loss function.

Table 5.5: *(L) Summary of Tested Parameters, (R) Summary of Top 5 Combinations with High Validation Accuracy for ResNet50 after 5 Iterations: Stage 2*

Parameter	Values
Learning Rate	[0.001,0.01,0.1]
Batch Size	[60,70,80]

	Learning rate	Batch Size	Train Acc	Val Acc
1	0.01	60	0.333	0.304
2	0.001	60	0.351	0.303
3	0.001	70	0.315	0.295
4	0.1	70	0.317	0.290
5	0.01	70	0.324	0.282

The introduction of learning rate decay has improved the learning compared to the model trained in absence of decay. However, it suffers even more from overfitting with slower learning speed, shown in Table A.3 from Appendix. The experiments showed that batch size is suitable to be kept smaller than its max-

imum capacity in training and decay would not be suitable for but moderately better with or RMSProp, which showed less overfitting behaviour.

Motivation of Fine-tuning

To deal with the bias caused by response class imbalance, class weights are introduced. These are included at model fitting, where the model is configured to give higher weights on the less populous class. Deriving class weights is done by computing the proportion of the number of works in a given class in the training set.

The second architecture comparison is by performance of model under the same setting as for Table 5.1 but with class weights. Ultimately, adding class weights prevented models from learning, implying its interference with convergence. It has also reduced the learning speed despite more moderate overfitting.

Table 5.6: *Summary of metrics for models trained for 5 epochs : Class weights added*

Architecture	Train Loss	Acc	Val Loss	Acc	Difference
VGG16	9.23E-04	0.233	**2.425**	**0.280**	- 0.053
InceptionV3	1.10E-03	0.138	3.128	0.103	- 0.076
ResNet50	9.24E-04	0.231	2.453	0.247	- 0.056

Stage 4 involved a model training under the chosen hyperparameters with larger number of epochs to investigate its convergence behaviour. As before, top 3 layers are trained but up to 30 epochs.

Table 5.7: *Training Configuration for Training Pretrained Models*

Epochs	Batch Size	N trained	Optimiser	Learning Rate	Dropout Rate
30	64	3	Adam	0.001	0.3

The Figure 5.3 shows the outcome of training with a larger number of epochs. The outcome agrees with the argument of Lecoutre *et al.* in that training pretrained models with replaced top layers show overfitting, which leads to degradation of validation loss. This suggested that the initial observation made in the previous stage has continued as a trend (overfitting). This implies that training the model with many iterations is not an effective solution: this motivated

introduction of early stopping to get the optimal result of the single training and retrain. Additionally, the constant learning rate may not be appropriate for longer epochs and for layers below. Hence, this should also be taken into account in the series model training. Another factor that stimulated multiple training is model bias where single training without class weight have resulted in strongly skewed model on the classes with largest number of training examples. Ultimately, according to the previous experiments, the best model from early stopping has been used in the followed retraining. It is selected as the model with the minimal validation loss.

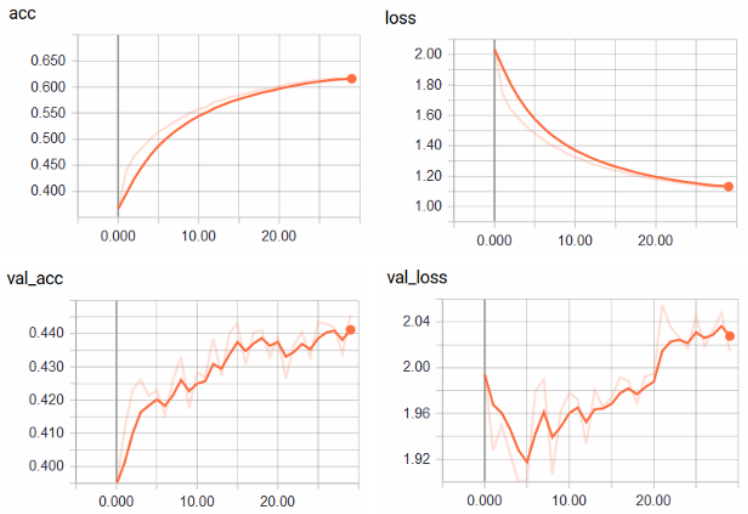

Figure 5.3: ***ResNet Model Training Monitored - Plots*** *: shows good optimisation on training data while it is resulted in overfitting with validation error degrading after reaching its local minimum at epoch 5. (generated by Tensorboard)*

Hence, using EarlyStopping regularisation, bottleneck training of top layers will attain a reliable accuracy model to proceed to fine-tuning.

Fine-tuning Strategy

As per Lecoutre *et al*, the number of trainable layers was increased. The first experiment shows the effect of increased number of top layers trained in a training instance from 1 to 10. The outcome shows that the more layers trained in a single instance, the better the model's loss value, but at the expense of greater overfitting and instability in optimisation. See Figure A.10.

This observation agrees with the results reported from the research that fine-tuned models outperform those with only top layers trained on the new task. In order to improve on its performance, *training by a series of retraining of layers in intervals is devised.* ***As per Yonsinski et al., combination of maximal transfer of weights (i.e. full pretrained model without top layer) and fine-tuning the deeper layers will be the strategy to follow.***

(1) Train top N layers without an addition of class weights to aid optimisation. Set the epoch value to be large (e = 15-20) with a large patience value (p = 5) for early stopping. The first stage promotes to fulfil maximum training capacity.

(2) When the lower layers from pretrained models are trained, top Dense (and Global Pooling layers) layers are trained with class weights subsequently. Having set the weights of previously retrained adjacent layers fixed, the retraining attempts to refine on the discriminative boundaries at this stage, propagating new features learnt to be included in classification. The epoch is set to be small (e = 5-10) considering quick learning and divergence observed in the previous experimentation result.

(3) Repeat the above by moving the interval of intermediate layers trainable further down and retaining or reducing the number of epochs and learning rate further.

    The number of layers to train at an instance is derived from the architectural design: ResNet is a collection of grouped blocks where layers in each block may hold "co-adopted features": experimental result conducted by Yosinski *et al.* on transfer learning specific layers showed that partially training any layers holding interactive features may degrade in performance due to inconsistency.[13] VGG16 is a sequential model where consecutive Convolutional layers are followed by a Max Pooling layer. It is adequate to incremental retraining, compromising large number of parameters in each layer.

(4) Evaluate the model on test set after training in the reversed order to avoid data miss-use.

5.3 Performance Metrics

In order to evaluate model performance, the following metrics are used. For some metrics where usual terminologies refer to binary classification problems, it is assumed as one vs all reformulation for multi-class problem. The test set from the full *wikipaintings_full* set is used for prediction. It contained over 8k images with around 330 image

per class with minimum of 96, median 202, and maximum of 1216 images. Hence, the predictive accuracy from the full test set may differ with the small test set which contains only 10 images per class.

Categorical Accuracy Top-1,3,5 Test Accuracy is reported in first instance at evaluating a trained model on a test set. This provides a holistic view as a quantitative generalisation capability measure.

Confusion Matrix A matrix summarising the outcome of test set evaluation per class. This is used as a main source of analysis tool of a model, for it providing per class mis-classification rate and enabling identification of class bias.

F1 Score A weighted average of F1 score (test accuracy) of each class.[30]

Precision Alternatively known as Positive Predicted Value (PPV), it is the proportion of true labels amongst truly predicted labels.[6]

$$\frac{TP}{TP + FP} \tag{5.1}$$

Recall Alternatively known as Sensitivity and True Positive Rate, it is the proportion of true labels predicted as true. [6]

$$\frac{TP}{TP + FN} \tag{5.2}$$

where TP (True Positive), TN (True Negative), FP (False Positive) and FN (False Negative).

Considering that it is a multi-class problem with over 20 classes, more weight is given to Test Accuracy and confusion matrix evaluation, following the general reporting fashion from research. Additionally, model behaviour during training is analysed. The correlation of training loss with validation loss and training accuracy and validation accuracy help identifying its learning capability and any sign of under- or over-fitting.

5.4 Model Training Results and Analysis

ResNet50

Combining the conclusion drawn from the previous section, ResNet50 model is trained with the base configuration as defined from Table 5.7 - as a way of validating the conclusions. Every retraining is recorded and it is summarised in Table 5.8. The

learning rate is kept to be constant since the assumption was that since it was sufficient for the top dense layer the upper layers which should have distinct features to object recognition, the learning rate of 0.001 would be small. The Phase 4 has shown that

Table 5.8: *Log of Training ResNet-tuned Model*

Phase	Layers Trained (index)	LR	Weight	Val Loss	Val Acc (%)	At Epoch	Top-1 Acc (%)
1	164 - 178 (top block (3 Conv) + 2 FC)	0.001	N	**1.979**	46.6	5	43.8
2	176 - 178 (2 FC + dropout)	0.001	Y	2.288	46.4	2	43.6
3	159-163 (half of penul. res block)	0.001	Y	2.189	48.0	4	46.3
4	156-158 (rest of penul. res block)	0.001	Y		15.0	-	-
5	130-160 (penul. res block + the preceeding block)	0.00001 + 0.1 factor decay	Y	2.148	51.0	1	48.4
6	120-130 (lower block)	0.00001 + exp. decay	Y	**2.090**	51.7	6	**49.1**
Total	(4 Residual blocks + 2 FC)						

an interval layer learning at this stage was not adequate - residual block has been split between the learning phases. This could also be explained by large learning rate and according to Yonsinski *et al.*s' definition of "fragile co-adapation" of weights learnt from two different tasks.[13] However, after 3 learning phases, the model has shown a very promising result on test accuracy. The learning phase it goes through can be explained qualitatively: The Figure 5.4 shows the generalisation ability of intermediate models having finished all the phases. As shown, class weights included from Phase 2 have effectively made corrections to the learnt features to adjust the output class boundaries: bias towards classes with large examples such as Impressionism, Realism have been moderated. Despite moderation, however, the effect persists in the final model for class Impressionism which is the class with the largest number of examples.

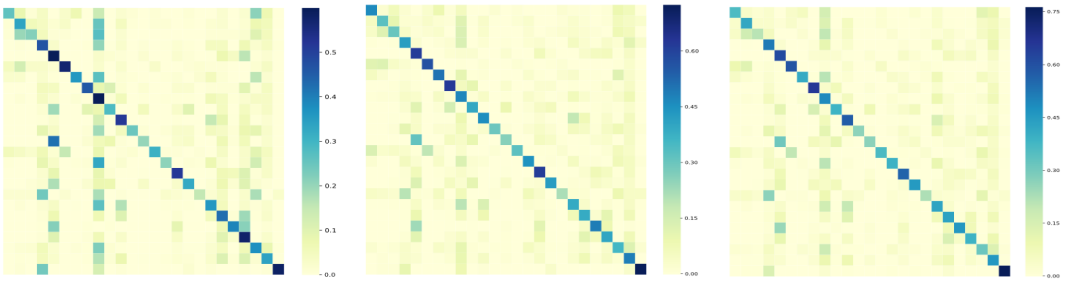

Figure 5.4: ***Confusion Matrix for Intermediate ResNet Models trained*** *: (Phase 1-3 from Left) intermediate models are tested to understand how model is refined over a series of training.*

Another observation made is that although adjustments from class imbalance were made to some extent, this was at the expense of slight drop in class accuracy for some

underrepresented classes.

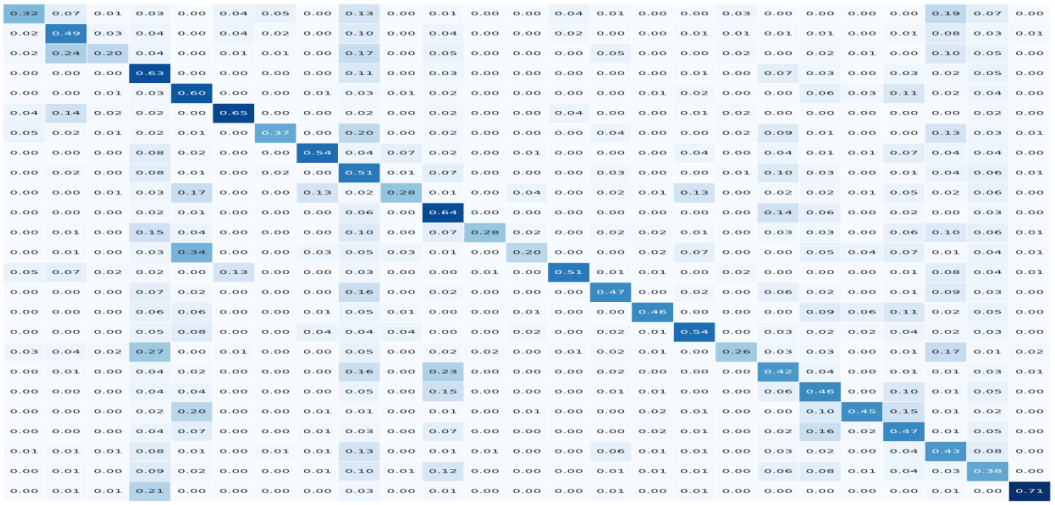

Figure 5.5: ***Confusion Matrix for Phase 6 ResNet Model trained*** *: it is predicted on wikipaintings_test in the full data set with 49.1% Top-1 Accuracy.*

The learning phases have shown that training the lower layers have increased the general accuracy. From Phase 3 to 6, some under-represented classes have improved however, model consistently mis-classifies in the major areas where classes have images that are visually similar. See bottom left corner of confusion matrices in Figure 5.4 and 5.6.

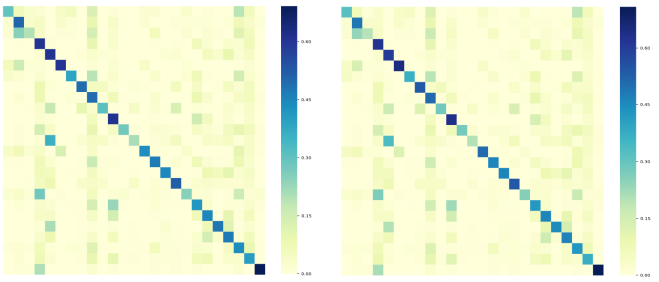

Figure 5.6: ***Confusion Matrix for Phase 5-6 ResNet Models trained*** *: (Phase 5-6 from Left) models are tested to understand how model is refined over a series of training.*

In comparison to the model built by Lecoutre *et al.* which attains 60.1% by fine-tuning the pretrained model, there is still more training to be done on this model. One improvement would be amending the model design by removing the penultimate Dense

layer and decreasing learning rate further for lower layers; with the interval of layers trained to match with the residual blocks. **The fully labelled confusion matrix of model from Phase 6 is available in Figure A.13.**

Various combinations of learning rate with decay are attempted while finding an optimal to retrain these layers were not found at the end of the project.

VGG16

VGG16 model trained consists of a base pretrained model, Global Average Pooling layer and a dense layer for output. Only one dense layer is added in this model in order to minimise the overall number of parameters. In comparison to ResNet50 where dropout was added in a hope for better generalisation in the *long term*, it was excluded in this model design in align with the testing output from Table 5.3. Whenever the loss value diverged at one training with a given learning rate, it is dropped by factor of 10 or further. See Figure 5.9 for training summary.

Table 5.9: *Log of Training VGG-trained Model*

Phase	Layers Trained (index)	LR	Weight	Val Loss	Val Acc (%)	At Epoch	Top-1 Acc (%)
1	top 3 (2 x pooling + 1 FC)	0.001	N	1.886	42.6	9 (10)	40.3
2	17 block3-conv3 (last Conv layer)	0.001	N	1.668	47.5	2 (5)	46.2
3	16 block-conv2 (penul. Conv layer)	0.00001	Y	1.627	48.2	1 (5)	46.4
4	16 block3-conv2 (penul. Conv layer)	0.00001	Y	1.608	49.7	5 (10)	48.1
5	15-16 block3-conv1,2	0.000005	N	**1.558**	51.8	1 (5)	49.9
6	15-16 block3-conv1,2	0.000003 (decay)	N	1.567	52.0	1 (5)	**50.3**
7	13-14 block2-conv3	0.000001	Y	1.600	50.9	1 (5)	-
Total	4 (3 Conv + 1 FC)					(45) 35hrs	

The Figure 5.9 shows that the first 3 phases are significant in order to remove class imbalance. The class weights added in running of Phase 2 removed large number of bias towards notable classes on the right.

Phases 4 to 6 were the steps to improve the overall accuracy given the effect from class imbalance is moderated. The improvement in model accuracy required substantially small learning rate as experimented, as more intermediate layers of the model are trained, which was a similar adjustment required for the previous model.

After Phase 6, the model has not improved even with a very small learning rate, and it indicated that further exploration of parameter should be required to promote further learning. The final model gave 50.3% accuracy for the test set with 78.4%, 88.3% for Top-3 and 5 respectively, which are competent results considering that only a third of total number of convolutional layers in the model are trained.

Lower learning rate for deeper layers can be explained by that features from mid-level layers from the tasks are very closely related. From the table it shows that *a relatively larger learning rate sufficed to train the weights in the higher level layers due to the disparity in high-level domain specific features: it adopted to the new data and provided with good generalisation result.*

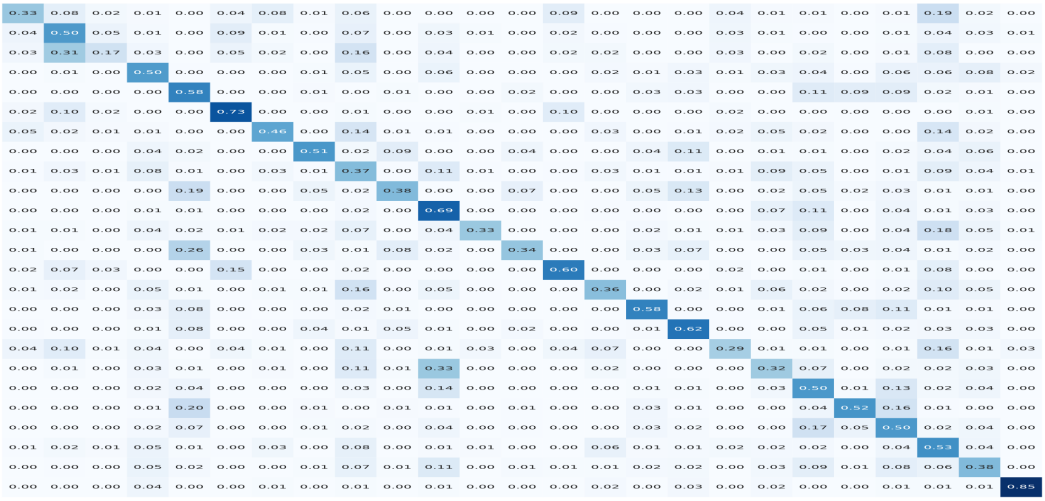

Figure 5.7: ***Confusion Matrix for VGG16 Model trained*** *: it is predicted on wikipaintings_test in the full data set with 50.3% Top-1 Accuracy.*

Additionally, later phases also show that continued retraining without class weights have reintroduced some bias which could have been improved otherwise. Although the final model has outperformed the model from Phase 4, the latter model outperforms the former in terms of individual class accuracy.

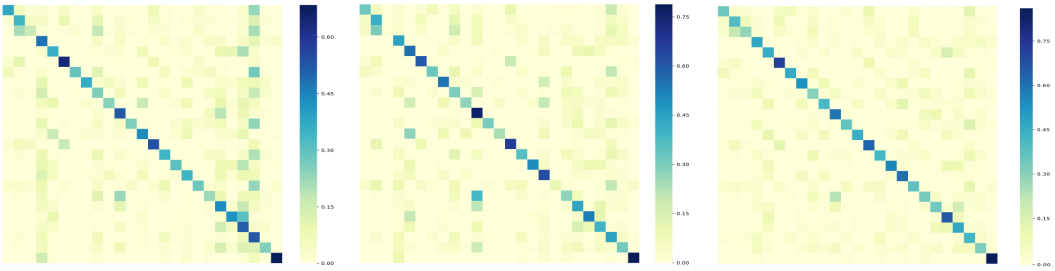

Figure 5.8: ***Confusion Matrix for First 3 Intermediate VGG16 Models trained*** *: (Phase 1-3 from Left) intermediate models are tested to understand how model is refined over a series of training.*

Hence, the model selection will be depending on which factor to prioritise between overall and individual class accuracy.

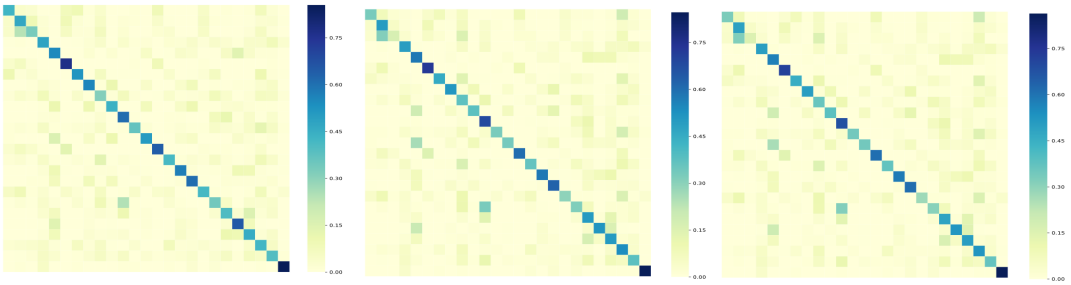

Figure 5.9: ***Confusion Matrix for Last 3 VGG16 Models trained*** *: (Phase 4-6 from Left) intermediate models are tested to understand how model is refined over a series of training.*

The fully labelled confusion matrix of model from Phase 4 is available in Figure A.12. The major areas of mis-classification are where two classes have visually similar characteristics due to its similar brushwork (Impressionism, Post-Impressionism), overlapping art movement periods (Baroque, Rococco, Mannerism, Renaissance), similar unique subject matter (Surrealism, Magic Realism), similar as inclusive classes (Abstract Art, Abstract Expressionism, Art Informel, Colour Field Painting., Cubism..)

For both models, *increasing number of layers tuned on the new data gave improved accuracy.* Reducing the learning rate for preceding layers contributed to this where the learning rate which was suitable for top layer didn't perform consistently well.

Figure 5.10: Painting used for Activation Map Generation - *Sunflowers* (1888) Vincent van Gogh

By comparing the two architecture designs and their performance, it is observed that classes of particular mis-classifications made differ except the areas which suffered from class imbalance: see Tables A.4 and A.5 Recall values - ResNet model is particularly biased towards Baroque style with other classes to which it is related. It has a variety of different visual elements featuring in Baroque as it was common across different countries with more individuality. The early works of Baroque share characteristics from Renaissance where majestic forms with vibrant colours were being used. VGG16 model is particularly biased towards classes which have character-

istics such as illustrative, detailed, outlined shapes: over general terms, classes such as Art Nouveau, Pop Art, Surrealism, Northern Renaissance share such features.

Analysing the activation maps of filters in early convolutional layers helped in providing the explanation of this observation. Activation maps visualise the area of the input image to which filters are activated, where every square represents the activation map for a filter randomly selected in the layer for visualisation - see Figure 5.11 which shows what the model learns as the most generic features. It is apparent that many filters for VGG16 capture lines outlining the shapes to finest degree in every angle. Later blocks of VGG16 shown in Figure A.11 suggests that deduction of shapes are largely derived from the features learnt from the previous layers to identify an object. ResNet on the other hand have filters capturing holistic shapes of a subject in a painting and being more responsive to tone. This observation shows that *the underlying generic features the models learn for object recognition were encoded differently and influenced the robustness of the derived model for style classification where all degrees of generic visual features are required to learn various classes.*

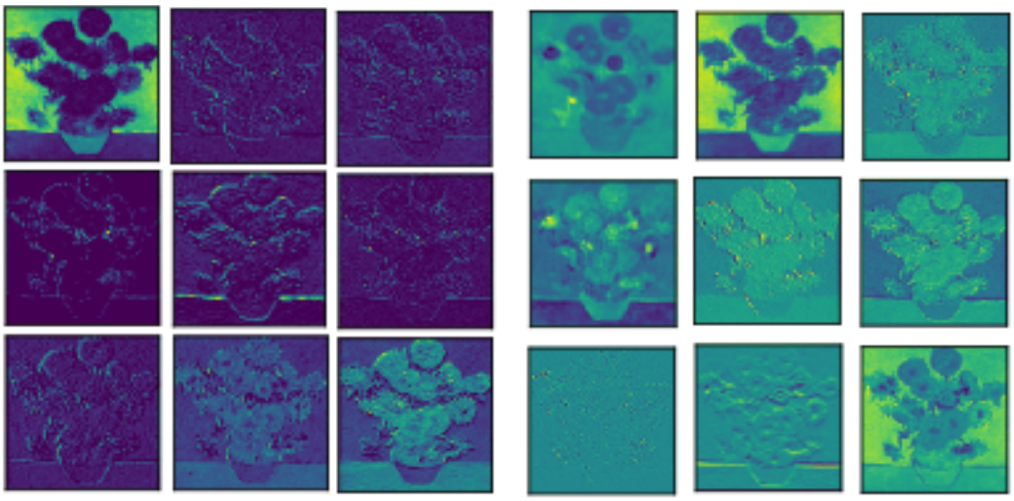

Figure 5.11: ***Activation Maps Generated from First Convolutional layers of VGG16 (L) and ResNet (R) Models***

Predictive Ability

An insight into general prediction ability is tested by passing 7 paintings which were not included in wikipaintings data but in *Image Database*. Further details are available in a supplementary material (Model Prediction Report).

- Given a painting with a style not constituting the class labels, models do guesswork to predict classes they are most likely to be biased to, from class imbalance. (Analytical Realism painting by Pavel Filonov) : ResNet [Post-Impressionism, Northern Renaissance..], VGG [Northern Renaissance, Art Noveau, Baroque..]

- Model specific bias help in getting high precision for the specific classes where the model is confident in its prediction. (ResNet model given Dutch Still Life paintings of 17^{th} century predicted the style as Baroque with more than 99% certainty.)

- For works belonging to classes that share similar visual features with discriminating factors derived from contextual information, models have performed less well. For example, Primitivism is not strictly a style but an approach taken by artists in their choice of subject matter and execution which influenced other style such as Expressionism. Hence, there is no apparent borderline for the models to discern between the two.

(a) *Still Life, Casket, Cup, Apples and Glass* (1909) Pablo Picasso

(b) *Portrait of Daniel-Henry Kahnweiler* (1910) Pablo Picasso

Figure 5.12: ***Paintings of Cubism Tested*** *: (L) Both models predicted correctly with 94% certainty, (R) ResNet model topped Abstract Art (60%), Abstract Expressionism (24%), VGG model topped Art Informel (35%) and Abstract Expressionism (23%)*

- As a model derived from pretrained models for object recognition, it is strong in making accurate predictions on works consisting of objects. Additionally, it is shown that it makes better prediction on works with similar compositions as those used as training examples. (Still Life painting by Pablo Picasso painted

under cubism: ResNet 93.9%, VGG 94.8%) The outcome can also be explained in terms of the style composition: cubist paintings of still life were prevalent, hence, the model should be very familiar with such composition but less likely on extreme geometric distortion in the depiction of facial features in a work with less common predecessors.(ResNet: Abstract Art 59.5%, VGG16: Art Informel 35%) Nonetheless, both models have predicted the work to be a work under Abstract Art in the generic term, implying that they are competent in the feature discrimination but still under-perform to distinguish fine-grained differences of sub-classes of Abstract Art.

- Models are found to be considerably sensitive to colour since large proportion of its confidence in its prediction are found to be derived from eccentric colour palette of some classes.

Chapter 6

Conclusions

Summary

The project aimed to deliver an contextual and empirical analysis on training CNN models built via transfer learning for fine-art paintings classification problem using pretrained models from object recognition. By building a flexible empirical framework extended from research specific design by Lecoutre *et al*, parameter and model fine-tuning are conducted more efficiently. The experimentation involved parameter tuning and exercise on the use of different architecture, resulting in models with competent generalisation ability given the space and time constraints. The contextual analysis on the nature of style classification problem and state-of-the-art model performance helped understand the level of generalisation ability in relation to the level-wise feature relationship between two tasks.

The experimentation process has derived a fine-tuning strategy of retraining pre-trained baseline models by increasing the number of layers trained from above incrementally/under a given interval. Two tasks are found to be very similar in terms of features learnt in low to mid level layers as expected where little adjustment was required for the layers. The resulted models of two different architecture types showed promising results with a large potential to develop further provided with more experimentation and tuning. Analysing the model performance both quantitatively and qualitatively identified model bias propagated from class imbalance, from classes with similar visual characteristics difficult to discern, and from the visual features learnt as "generic" in the bottom most hidden layers: this also gave an insight to how models identify objects from object recognition task.

Future Work

With the majority of initial objectives of the project covered, there has been some observations made with insightful conclusions. Possible future work would be:

- Extending the models produced in this project to improve its accuracy which will involve further parameter search and retraining.
- Extending the proposed 2-stage model building process using autoencoders for formal comparison of two models - explained in Appendix .
- Extending the use of cross validation in training for more accurate training result. Extending pre-processing steps prior to training such as cropping to maintain original resolution of input image and minimise information loss from distortion. The latest research by Rodriguez et al. in Dec 2018 proposes a 2-stage learning of CNN models on individual image patches followed by model concatenation process by training output data on a shallower CNN. The best model achieves 77.5% accuracy on 19 styles classification which has overtaken the latest state-of-the-art model.[24]
- Revising the existing data set to build more coherent class structure. As advised from the domain expert, the chosen *wikipaintings* data set had few incorrectly labelled art works from WikiArt. This suggests that the closer collaboration with the domain expert will be required in research if a competent model that *correctly* predicts the style of a work to have a practical use.

Tuning a neural network model for a classification task is considered to be *more an art than a science.* Every degree of change in one component may generate drastically different results but when the optimum is found, its classification ability would be extremely powerful. As the model attempts to capture imagination from the creativity in art[20], it makes use of the known (visually distinguishable features extracted for object recognition) to deduce the unknown state, optimal for style classification model. The model itself is a piece of art which in the end could embody the features shaping visual aesthetics of centuries. Further research for this problem will take a step closer to full automation, outperforming the human decision making process.

Glossary and Abbreviations

Glossary

Image Descriptors a vectorised representation of image features extracted from extraction algorithms, capturing visual qualities of an image such as texture and intensity. It is formally defined as "a pair,feature vector extraction functionand-distance function,used for image indexation by similarity."[31]. 20

PiCoDes a type of image descriptor as a later-developed binary image representation learner. It is especially powerful in learning image categories beyond the initial categorisation due to its ability of good generalisation.[32]. 20

Abbreviations

Adam Adaptive Moment Estimation. 63

CNN Convolutional Neural Networks. 9, 10, 12, 13, 38

ILSVRC ImageNet Large Scale Visual Recognition Challenge. 19

ResNet Residual Neural Networks. 21, 66

ROC Receiver Operating Characteristics. 67

SGD Stochastic Gradient Descent. 16, 45, 61

Appendix A

Gradient-based Optimisation Algorithms

Stochastic Gradient Descent (SGD)

Deriving the gradient of the output function is achieved by a well constructed statistical estimation computed over a small data set, which is a minibatch gradient descent, where one update will be done at each time step.[5] Taking a small batch to process has a benefit of faster convergence in comparison to "deterministic" gradient methods where the entire training set will be used to infer parameters once.[5] Taking a small batch size will also contribute as a method of regularisation, to deal with overfitting: noise generated as a result will contribute as additional learning required by the model.[5]

SGD algorithm uses an unbiased estimate of the exact gradient by random sampling of images in data set to get a minibatch of size m. The estimated gradient $\hat{g}$ is an average of gradient of per-example loss (L) for all $\mathbf{x}^{(i)}$ in the minibatch with the true label $y^{(i)}$:[5]

$$\hat{g} = \frac{1}{m}\nabla_\theta \sum_i^m L\left(f(\mathbf{x}^{(i)}; \theta), y^{(i)} \right) \tag{A.1}$$

Every weight at the end of k^{th} iteration will be updated using $\hat{g}$:

$$\theta \leftarrow \theta - \eta \cdot \hat{g} \tag{A.2}$$

where η_k is the learning rate: the step size of the model to reach the minimum.[5]

The decay of learning rate is required for SGD optimiser to guarantee convergence since the noise generated from the random sampling for minibatch to derive the estimate gradient will be retained in the neighbourhood of the minimum.[5] Without a reducing learning rate, minimum will not be achieved.

An extension is the addition of a momentum term for acceleration in the convergence

of the optimiser.[5] It is shown that the effect of momentum is comparable to that in physical particles such that its inclusion allows reduced irregular oscillations near the minimum and a wider range of values for learning rates leading to convergence.[33] This will modify the equation (A.2) to

$$\upsilon \leftarrow \alpha\,\upsilon - \eta \cdot \hat{g} \tag{A.3}$$

$$\theta \leftarrow \theta + \upsilon \tag{A.4}$$

With υ as the point of direction for parameter update in its scope, the implication of the momentum is the exponentially decaying contribution of previous gradients to update the current parameters.[33][5] Essentially, the learning rate decay is not necessary when convergence is guaranteed by decaying gradient. The term "α" determines the decay rate.

The initial learning rate and the decay factors are hyperparameters to be tuned by experimentation.

Adaptive learning rates

Another type of practical algorithm implements a varying learning rate for every weight term at each time step. This specialisation aims to deliver "domain-specific feature weightings" such that features can be treated differently depending on their importance and the frequency in their appearance through the training process.[34]

The parameter is updated with an individually configured learning rate, derived by scaling the global learning rate to be "inversely proportional to the square root of the sum of all historical squared values of the gradient"[5] as in Eqn. A.5 and A.6 This ensures that weight terms with small or large gradients can be treated differently where parameters with large partial derivatives should have a drop in subsequent learning rate, while small values will see a mild increase in the learning rate.[5] The algorithm is named as *AdaGrad*:

$$r \leftarrow r + \hat{g} \odot \hat{g} \tag{A.5}$$

$$\theta \leftarrow \theta - \frac{\eta}{\delta + \sqrt{r}} \odot \hat{g} \tag{A.6}$$

with estimated gradient $\hat{g}$ as in Eqn. A.1 where the equation (A.5) represents the sum of products ($\odot$) of previous gradients by defining an intermediate variable r which accumulates the gradients up to the current iteration. Parameter update applied in

(A.6) shows the revised learning rate where a small constant term δ is introduced for the purpose of stable numerical division.[5]

There exists a potential defect in the accumulation of squared gradients for obtaining reliable learning rates[5]: it would produce learning rates of decreasing magnitude. Ultimately, the significant drop in the learning rate before its arrival in the convex region for convergence would lead to a halt in learning.[5].

An extension of *AdaGrad* is *RMSProp* which circumvents the problem by replacing the accumulation with "exponentially weighted moving average".[5]

$$r \leftarrow \rho\, r + (1-\rho)\, \hat{g} \odot \hat{g} \tag{A.7}$$

$$\theta \leftarrow \theta - \frac{\eta}{\sqrt{\delta + r}} \odot \hat{g} \tag{A.8}$$

The equation (A.7) shows that the value ρ is introduced, which is the proportion for weighting running average of the magnitude of gradient of the previous and the current.[35]

The last optimiser of interest with adaptive learning rates is Adaptive Moment Estimation (Adam).[36] It is an enhancement of *RMSProp* by combining with momentum;[36]

$$\begin{aligned} s &\leftarrow \rho_1 \ + (1-\rho_1)\, \hat{g} \\ r &\leftarrow \rho_2\, r + (1-\rho_2) \hat{g} \odot \hat{g} \end{aligned} \tag{A.9}$$

$$\begin{aligned} \hat{s} &\leftarrow \frac{s}{1-\rho_1^t} \\ \hat{r} &\leftarrow \frac{r}{1-\rho_2^t} \end{aligned} \tag{A.10}$$

$$\theta \leftarrow \theta - \eta \frac{\hat{s}}{\sqrt{\hat{r}} + \delta} \tag{A.11}$$

where s and r denote first- and second- order moment estimations respectively, and bias correction in Eqn. (A.10) allows the model to maintain low bias value throughout training in contrast to *RMSProp*.[5] It is the most robust first-order optimisation algorithm according to literature which resolves the issues discussed including, vanishing learning rate, slow convergence and high variance.[36][5]

Alternative optimisation methods

The second order optimisation algorithms are also applicable where it uses the second order partial derivatives, represented by Hessian matrix[12], which takes into

account the curvature of the error surface, help avoid model being idle in the local minima.[12]

The definition of the **Hessian** matrix is:[5]

$$H(f)(x)_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} f(x) \tag{A.12}$$

where $f(x)$ is the function to optimise. Using the second derivatives, at every iteration, Taylor's expansion to second order will generate:[5]

$$f(x) \approx f(x^{(0)}) + (x - x^{(0)})^T g + \frac{1}{2}(x - x^{(0)})^T H (x - x^{(0)}) \tag{A.13}$$

where g denotes the first derivative (gradient) and $x^{(0)}$ represents the current point.[5] An estimation to the new point $x^{(0)} - \eta\, g$ can be derived as:[5]

$$f(x - \eta g) \approx f(x^{(0)}) - \underbrace{\eta g^T g}_{\text{improvement by gradient}} + \underbrace{\tfrac{1}{2}\eta^2 g^T H g}_{\text{correction from curvature}} \tag{A.14}$$

where the directional second derivative with a unit vector d is $d^T H d$.[5] Hence the revised learning rate would be:

$$\eta^* = \frac{g^T g}{g^T H g} \tag{A.15}$$

Thereby it is more efficient in finding a correct direction to proceed in comparison to the first order methods.

Newton's method is an example which requires an exact Hessian matrix for this derivation. Practical implementation of optimisation algorithms utilises first order algorithms, however; due to larger gain in time and complexity for derivation.[5] Quasi-Newton Method takes this into account and makes an estimation to the inverse of Hessian using the first order derivatives. *Broyden–Fletcher–* Goldfarb–Shanno (BFGS) formula is one way of making estimation updates of the matrix at the end of each iteration.[12]

Architecture Analysis

VGG

Simonyan *et al.* proposed an enhanced sequential CNN model inspired from AlexNet. This was accompanied by an experimental approach to understand the implication of an increased depth of the model. The result reported that deeper models outperformed the shallow by nearly doubling the number of layers from AlexNet.[27] In order to minimise the number of parameters from the increased number of layers, and maintaining

satisfactory discrimination ability of convolutional layers on nonlinear functions equally, VGG architecture is comprised of convolutional layers with small 3x3 filters only with increased number of feature maps.[27] The use of multiple convolutional layers with smaller filter size provides equivalent power as one 7x7 convolutional layer.[27]

This is a typical case exemplifying the start of the trend in the design of convolutional neural networks to build a competent model: increasing the number of layers in the model, along with the number of units of each layer: both depth and width[37][4], disseminating feature extraction abilities in more fine grained layers.[27] Despite of such improvement, the architecture has limitations such as large number of parameters downgrading its computational efficiency for optimisation and making it "prone to overfitting" for a limited training data.[4] This can be mitigated by taking small batch size and limiting the trainable layers at one instance.

Inception Module Architecture

The first version of inception architecture was the winner of IRSVRC 2014 competition for classification task with VGG as the runner-up, formally named as GoogLeNet (InceptionV1).[4] The key characteristic is deviation from original sequential model, but a modular network structure; stacked so called *Inception* modules to build more sparse architecture, as a way of addressing the arising problems of having more densely connected architecture.[4][38]

It utilises 1x1 filter convolutional operation to alleviate computational effort of proceeding 3x3, 5x5 convolution operations, and dividing the representation to learn underlying compact information, "embeddings" in the data.[4] See Figure A.5 for the module design.

The later revised model is Inceptionv3, which is tested in this project, with improved generalisation performance from the previous version.[29] The key refinement made in this version is having variations of inception modules by dividing convolutional layers with smaller filter sizes - as implemented in *VGG*.[29] Auxiliary classifiers are reinforced to contributes to overall prediction by probability output in the intermediate stages of the network: this was motivated from the initial design of GoogLeNet where they found intermediate layers capture features with high discriminative power.[4][29]

Hence, it implies that the architecture is designed to be optimal with trained data. The point of experimentation would be on how well such design would contribute in transferability of the generalisation power to the target task. In terms of its training, the sparse architecture allowed the parameter reduction down to around 20% of that of VGG.[4] This is achieved by dropping dense layers in the top of the model, replaced with concatenation.[29]

Table A.1: *Summary of Performance of Key CNN models on ImageNet Classification (as implemented in Keras)*

Architecture	Accuracy (%)	Top-5 Error (%)	Params(M)
VGG16	71.20	8.83	138.4
InceptionV3	77.90	6.82	23.9
ResNet50	74.90	5.04	25.6

ResNet

The Residual Neural Networks (ResNet) is a model which is built with a series of building blocks consisting of shortcut connections of layers bypassing subsequent 1-2 convolutional layers, which are later merged at the activation function.[9] This short cut connection is an identity function such that given a underlying nonlinear mapping $H(x)$ defined to be a sum of identity function with residual function $F(x)$, $H(x) = F(x) + x$, it is redefined as $F(x) = H(x) - x$ such that the model to learn the residuals instead.[9]

Identity block design is shown in Figure A.6. The results reported that the ResNet models generalise well with no degradation with increased depth, in comparison to plain sequential models, thereby ResNet variants have different number of layers: as many as 152-layer residual net.[9]

The Table A.1 summarises the general performance of aforementioned models on ImageNet classification, as implemented using Keras[1]. Hence, there may be small discrepancies in the accuracy and error rate in performance metrics as discussed in the original research papers.

Table A.2: ***Summary of Latest CNN models from IRSVRC*** *: shows its performance on ImageNet classification task with the maximum number of layers constituting the architectural designs.[7]*

Architecture	Year	Error rate (%)	Maximum Number of Layers
AlexNet	2012	16.4	8
VGG	2014	7.3	19
GoogLeNet	2014	6.7	22
ResNet	2015	3.57	152

[1] https://github.com/keras-team/keras-applications

ROC Explained

Specificity Alternatively known as True Negative Rate, it is the proportion of true negative predicted as negative. In multi-class classification problem, this is interpreted as an observation not part of one class is correctly not classified as this class of interest. [6]

$$\frac{TN}{TN + FP} \tag{A.16}$$

ROC curve is a summary plot "assessing sensitivity and specificity for varying thresholds."[6] It is a collection of confusion matrix for different classification rules and plotted along the cut-off values. This is especially useful in model comparison and selection. Since high sensitivity and low 1-specificity is desired, ideally preferred plot should show the curve covering the majority of the area of the plane, closest to the top left corner.[6]

Quantitative measure is the AUC (Area Under the Curve) value of the curve, where in the scale of [0,1], 0.5 denotes classifier with no useful information with higher value denoting higher probability for the classifier to correctly classify cases.[6]

Autoencoder Training

Additional data from the School of Art History was initially intended for model training and prediction. However, in data preparation it was revealed that the images are not labelled with style. Manual annotation of over 30k would be challenging even with the help of domain expert.

Hence, another method of exploiting the data was designed. An **unsupervised learning** method of stacked autoencoders is devised by transferring the weights to build a new **supervised learning** model.

Considering that unlabelled data is common in real-life data, the proposed training methodology would have wide utility. In-depth exploration of this is beyond the scope of this project.

In brief, stacked autoencoders are one type of neural networks with a symmetric architecture design, such that the initial layers encodes an image while the top layers beyond the layer of symmetry decodes the compressed form, in a hope of recreating the original input.[11]

1. For layer-wise training, N Autoencoders are built each with 2-3 Convolutional layers mirroring those from VGG16 architecture.

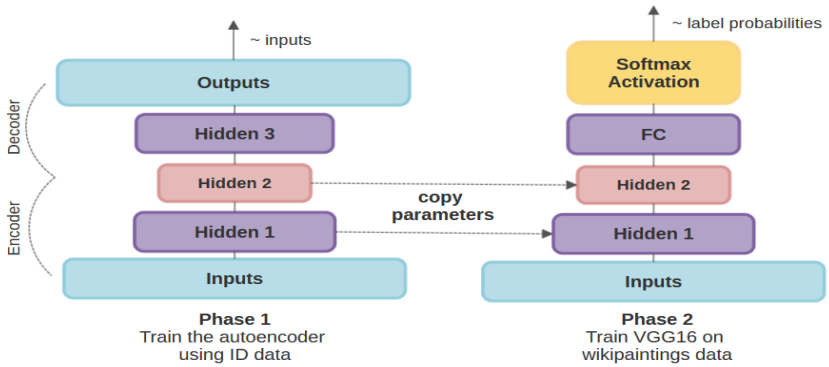

Figure A.1: ***Use of Unsupervised Learning for Model Building with Autoencoders[11]*** *: shows the process described below;*

2. They are individually trained on the training set generated from ID dataset. (ID dataset is split in 8:1:1 ratio.) The chosen loss function is *mean squared error* whereby the image can be learnt for the model to be capable of reproducing it.

3. The learned weights from *Unsupervised Learning* phase will be copied to the new model at initialisation stage. The chosen architecture type for supervised learning is VGG16: its sequential design makes parameter transfer applicable.

Results

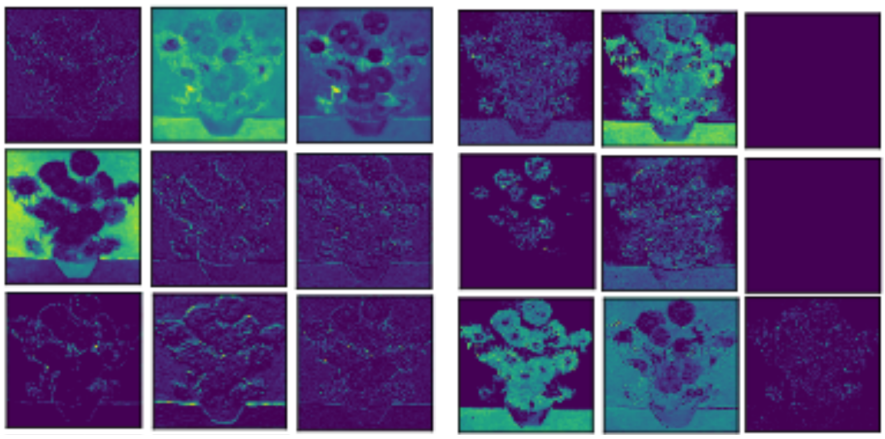

Figure A.2: ***Activation Maps Generated from First Convolutional layers of VGG16 (L) model and Trained Autoencoder (R)****: autoencoder learns similar features as those from VGG16 trained on ImageNet while further training will be needed for better comparison.*

The layer-wise autoencoder training has required a larger number of training than expected where full training of VGG16 model as an autoencoder was beyond the storage limit. Due to the extended experimentation for building the main model, only the lowest block of the model was trained with appropriate learning rate. The batch size was set as 15 which prolonged the training time further. Having trained and copied the weights of the first layer, the new VGG16 model is trained in the same fashion as Phase 1 from Table A.5. It achieved 35% validation accuracy. The potential improvement would be building autoencoder for all the layers.

Plots

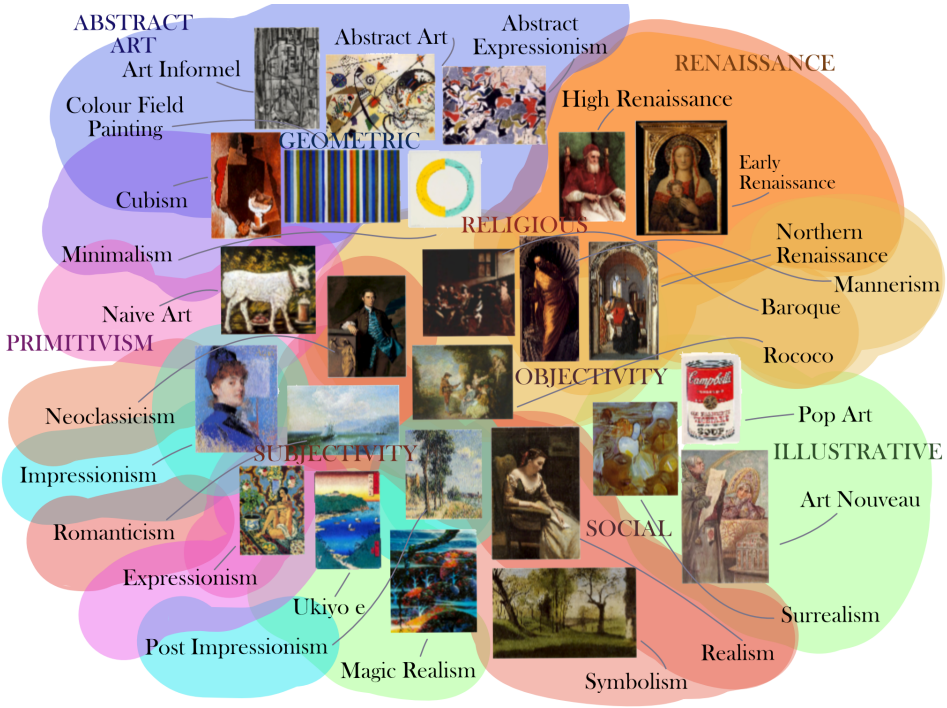

Figure A.3: ***Classes grouped by Common Factors*** *: the bubble chart shows each class with size representing the training set counts with common groups by colour.*

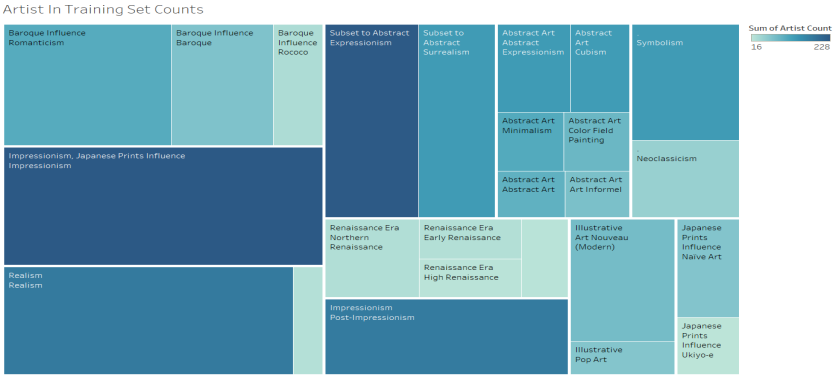

Figure A.4: ***A Heatmap of number of artists per class*** *: the bar chart grouped by number of artists per class where each class consists of works by several artists, with varying degree of variability. (The darker the hue, the more artists whose works present in the class. The larger the block, the greater the number of works present in the class.) Note that the number of artists does not correlate with the total number of works per class from Figure 4.3*

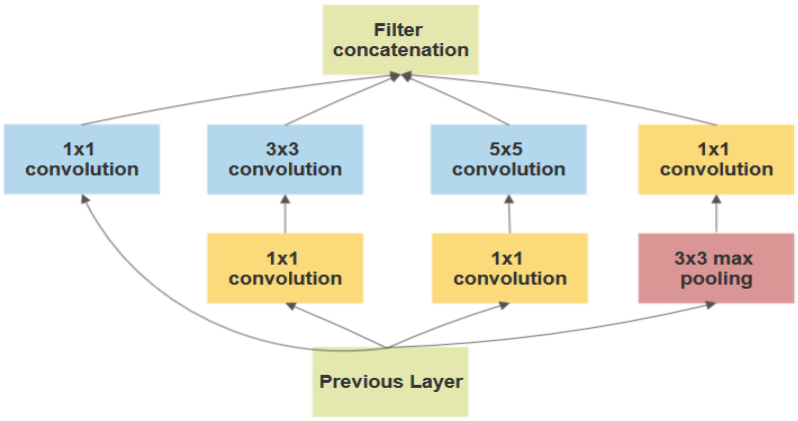

Figure A.5: ***A Diagram of an Inception Module in GoogLeNet Variants*** *: GoogLeNet architecture takes the form of stacked Inception modules.[4]*

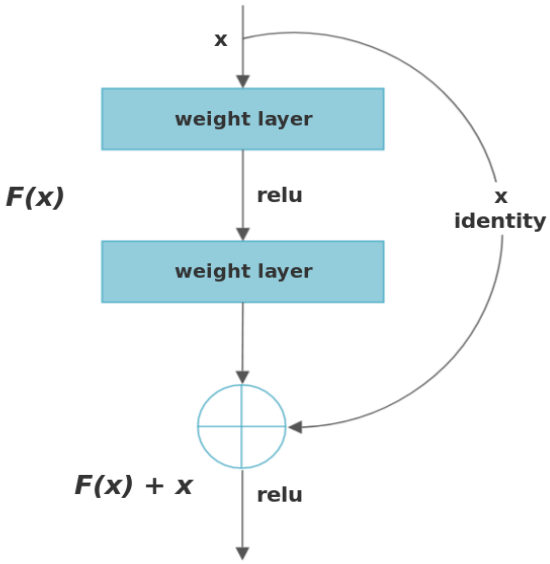

Figure A.6: ***A Diagram of a residual block in Residual Neural Networks (ResNet)[9]*** *: ResNet architecture takes the form of stacked residual blocks.*

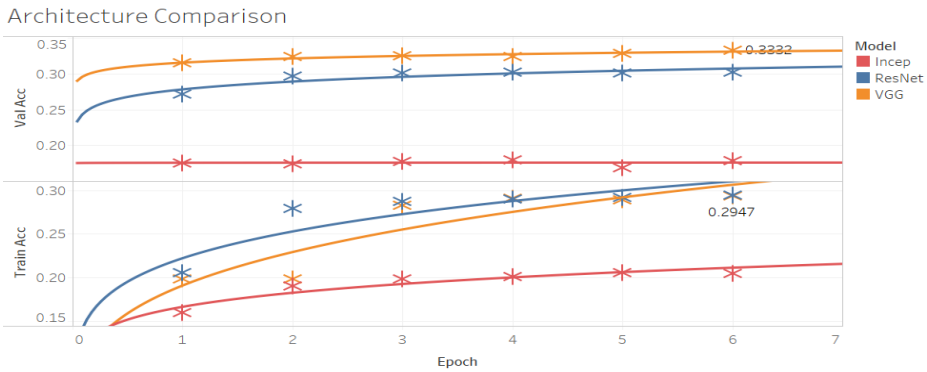

Figure A.7: ***Architecture Comparison***

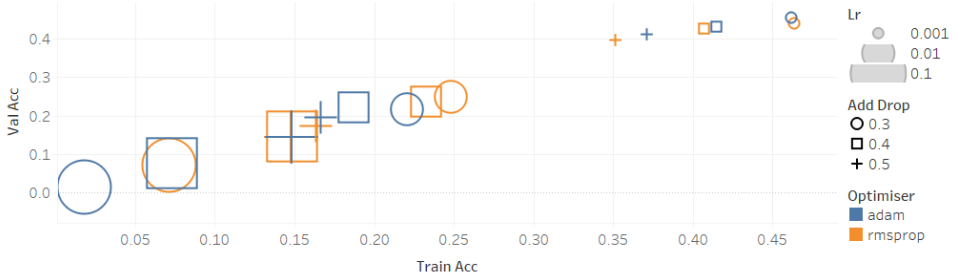

Figure A.8: ***Hyper-parameter Tuning for VGG16: Stage 1*** *: Learning rate, Optimiser and Dropout values are considered.*

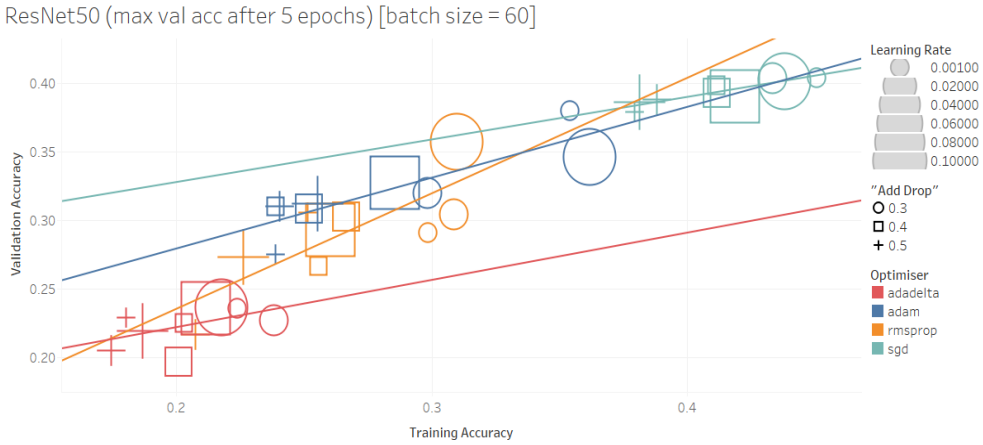

Figure A.9: ***Hyper-parameter Tuning for ResNet50: Stage 2*** *: Learning rate, all Optimiser types and Dropout values are considered.*

Table A.3: *Summary of Top 5 Combinations with High Validation Accuracy for ResNet50 after 5 Iterations: Stage 3 (Decay types with SGD)*

	Learning rate	Decay Type	Train Acc	Val Acc
1	0.1	exp	0.473	0.371
2	0.1	step	0.498	0.363
3	0.1	rate	0.403	0.328
4	0.01	step	0.349	0.304
5	0.01	exp	0.333	0.300

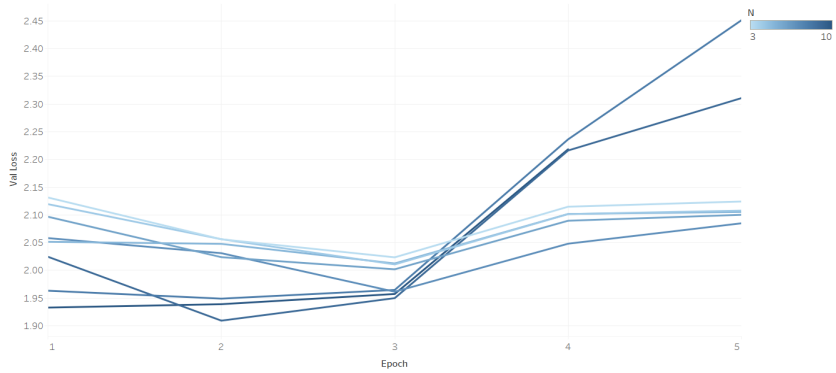

Figure A.10: ***The Effect of Increasing Trainable Layers (N) at one Instance on Validation Loss for ResNet*** *: recorded for 5 iterations. Despite of rapid convergence of training loss with increasing number of trainable layers at one time instance, validation loss diverges further after reaching its minimum for larger N.*

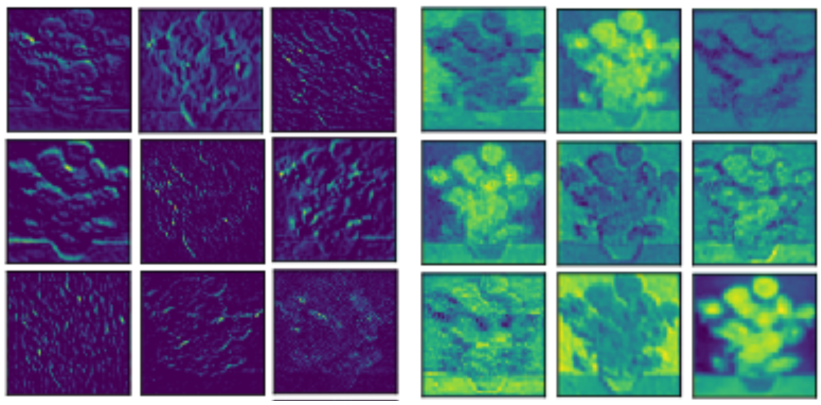

Figure A.11: ***AActivation Maps Generated from Convolutional layers of VGG16 (L) (4th layer) and ResNet (R) (2 layer in the 4th branch) Models****: as mentioned, higher convolutional layers have filters which are responsive to similar features as the lower. (Fine grained lines, holistics shapes and tone)*

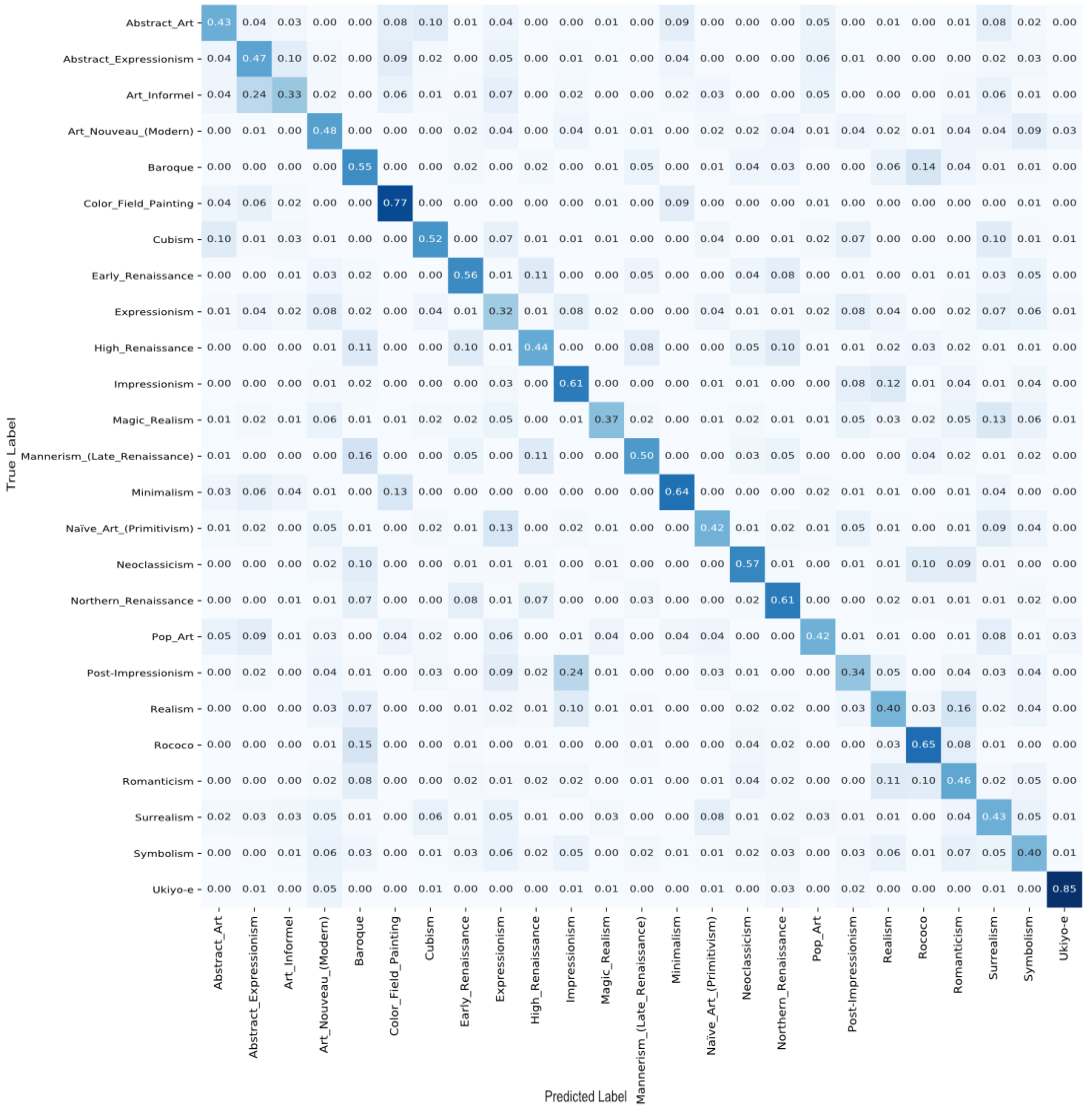

Figure A.12: ***Fully labelled confusion matrix for Phase 4 VGG16 Model****: major areas of misclassification are where two classes have visually similar characteristics due to its similar brushwork (Impressionism, Post-Impressionism), overlapping art movement periods (Baroque, Rococco, Mannerism, Renaissance), similar unique subject matter (Surrealism, Magic Realism), and similar as inclusive classes (Abstract Art, Abstract Expressionism, Art Informel, Colour Field Painting., Cubism..)*

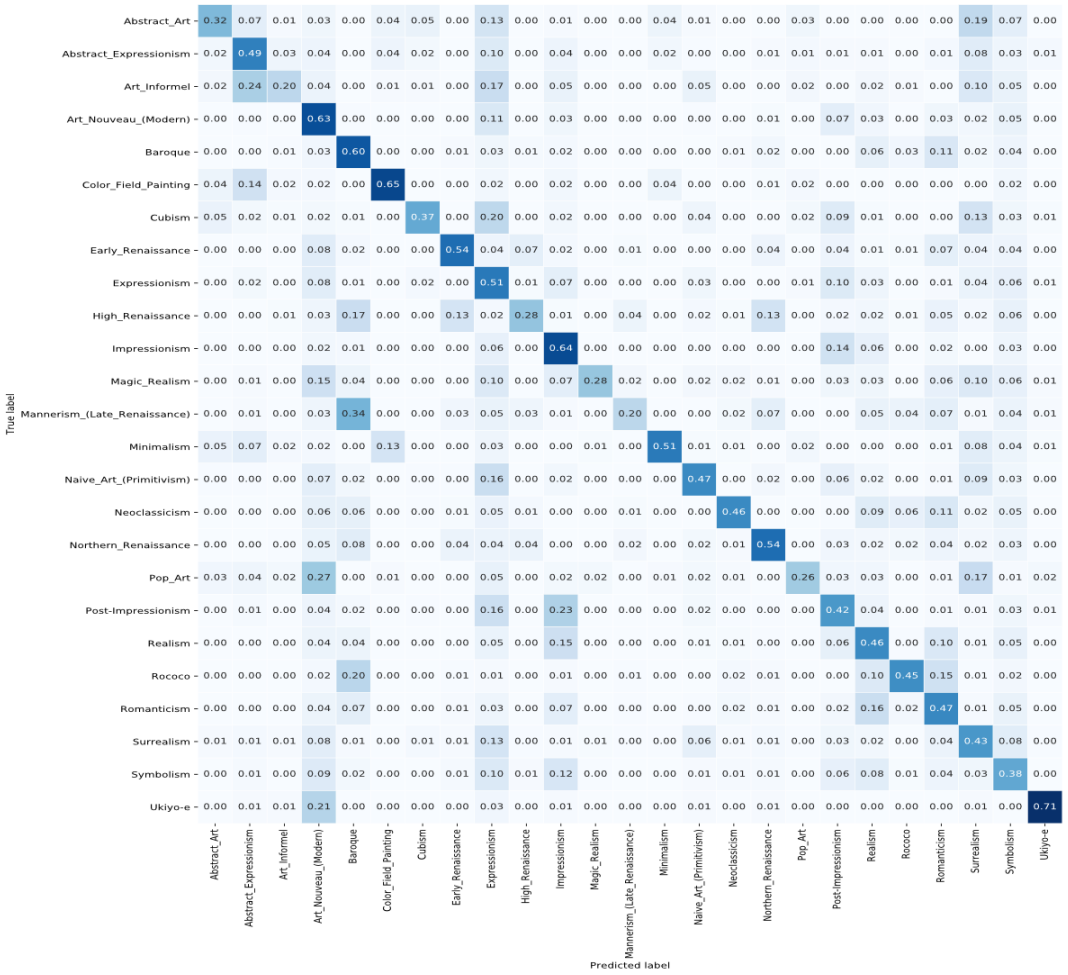

Figure A.13: ***Fully labelled confusion matrix for Phase 6 ResNet50 Model****: major areas of misclassification are where two classes have visually similar characteristics due to overlapping art movement periods (Baroque, Rococco, Mannerism), common line heavy qualities (Pop Art, Art Nouveau, Ukiyo-e) , similar as inclusive classes (Abstract Art, Abstract Expressionism, Art Informel, Colour Field Painting., Cubism..). It is more affected by class imbalance.*

Table A.4: *Precision and Recall Values for Phase 6 - ResNet Model*

Class	Precision	Recall	F1 score
Abstract Art	0.46	0.32	0.38
Abstract Expressionism	0.50	0.49	0.50
Art Informel	0.40	0.20	0.26
Art Nouveau(Modern)	0.39	0.63	0.48
Baroque	0.46	0.60	0.52
Color Field Painting	0.72	0.65	0.68
Cubism	0.65	0.37	0.47
Early Renaissance	0.56	0.54	0.55
Expressionism	0.34	0.51	0.41
High Renaissance	0.44	0.28	0.35
Impressionism	0.61	0.64	0.62
Magic Realism	0.62	0.28	0.38
Mannerism (Late Renaissance)	0.48	0.20	0.28
Minimalism	0.78	0.51	0.62
Naive Art (Primitivism)	0.48	0.47	0.47
Neoclassicism	0.72	0.46	0.56
Northern Renaissance	0.61	0.54	0.57
Pop Art	0.55	0.26	0.35
Post-Impressionism	0.38	0.42	0.40
Realism	0.56	0.46	0.51
Rococo	0.58	0.45	0.51
Romanticism	0.48	0.47	0.48
Surrealism	0.47	0.43	0.45
Symbolism	0.31	0.38	0.34
Ukiyo-e	0.77	0.71	0.74

Table A.5: *Precision and Recall Values for Phase 6 - VGG16 Model*

Class	Precision	Recall	F1 score
Abstract Art	0.42	0.33	0.37
Abstract Expressionism	0.46	0.50	0.48
Art Informel	0.31	0.17	0.22
Art Nouveau (Modern)	0.50	0.50	0.50
Baroque	0.47	0.58	0.52
Color Field Painting	0.62	0.73	0.67
Cubism	0.55	0.46	0.50
Early Renaissance	0.56	0.51	0.54
Expressionism	0.39	0.37	0.38
High Renaissance	0.44	0.38	0.41
Impressionism	0.61	0.69	0.65
Magic Realism	0.60	0.33	0.42
Mannerism (Late Renaissance)	0.50	0.34	0.40
Minimalism	0.67	0.60	0.63
Naive Art (Primitivism)	0.42	0.36	0.39
Neoclassicism	0.62	0.58	0.60
Northern Renaissance	0.56	0.62	0.59
Pop Art	0.41	0.29	0.34
Post-Impressionism	0.45	0.32	0.37
Realism	0.50	0.50	0.50
Rococo	0.44	0.52	0.47
Romanticism	0.45	0.50	0.48
Surrealism	0.45	0.53	0.49
Symbolism	0.39	0.38	0.38
Ukiyo-e	0.80	0.85	0.82

Table A.6: *Reference to Works Included in Figure 4.1 clockwise from Abstract Art : and for Prediction*

Class	*Title* (Date) Artist
Abstract Art	*The Ten Biggest, No 2* (1907) Hilma af Klint
Impressionism	*Flowers in a Crystal Vase* (1882) Edouard Manet
Analytical Realism	*Universal Flowering* (1916) Pavel Filonov
Art Nouveau (Modern)	*The Sunflower* (1907) Gustav Klimt
Post-Impressionism	*Sunflowers* (1888) Vincent van Gogh
Cubism	*Still Life, Casket, Cup, Apples and Glass* (1909) Pablo Picasso
Dutch Still-life (Baroque)	*Still Life with Fruit and Vegetables, with Christ at Emmaus in the background* (1630) Floris van Schooten
Post-Impressionism	*Apples* (1887) Vincent van Gogh
Post-Impressionism	*Basket of Apples* (1893) Paul Cezanne

Table A.7: *Reference to Works included in Figure 4.2 and A.3*

Class	*Title* (Date) Artist
Abstract Art	*Transverse Line* (1923) Wassily Kandinsky
Abstract Expressionism	*The Surge* (1958) Conrad Marca-Relli
Art Informel	*Jeanne d'Arc* (1944) Jean Rene Bazaine
Art Nouveau (Modern)	*Scribe* Sergey Solomko
Baroque	*Calling of Saint Matthew* (1600) Caravaggio
Colour Field Paintings	*Citadel* (1962) Gene Davis
Cubism	*The figure with fruit dish* (1917-1924) Pablo Picasso
Early Renaissance	*Madonna and child* (1450) Jacopo Bellini
Expressionism	*Decorative figure on an ornamental background* (1925) Henri Matisse
High Renaissance	*Portrait of Pope Julius II* (1511) Raphael
Impressionism	*Portrait of Alma* (1887) Carl Larsson
Magic Realism	*Toccata and Fugue* (1992) Eyvind Earle
Mannerism (Late Renaissance)	*A Philosopher* (1570) Tintoretto
Minimalism	*Split Ring Image* (2008) Robert Mangold
Naive Art (Primitivism)	*Easter Lamb I*, Niko Prosmani
Neoclassicism	*Thaddeus Burr* (1760) John Singleton Copley
Northern Renaissance	*The Annunciation, the wing of Polyptych of the Virgin* (1445) Dieric Bouts
Pop Art	*Campbellś Soup Can* (Old fashioned Vegetable) (1969) Andy Warhol
Post-Impressionism	*Quay at Dieppe* (1905) Gustave Loiseau
Realism	*The Letter* (1865) Camille Corot
Rococo	*The Lesson of Love* (1716) Antoine Watteau
Romanticism	*On the coast* (1875) Ivan Aivazovsky
Surrealism	*Mirror Women-Mirror Head* (1982) Salvador Dali
Symbolism	*View of the Villaborthese* (1858) Gustave Moreau
Ukiyo-e	*Bay at Kominato in Awa Province* (1853) Utagawa Hiroshige

References

[1] M. K. Cowan, "Tikz Package for Drawing Neural Networks." [Online]. Available: https://github.com/battlesnake/neural 6, 13

[2] P. Veličković, "2D Convolution." [Online]. Available: https://github.com/PetarV-/TikZ/tree/master/2D%20Convolution 6, 15

[3] "Transfer Learning using Alexnet MathWorks." [Online]. Available: https://www.mathworks.com/help/deeplearning/examples/transfer-learning-using-alexnet.html;jsessionid=6aae477eb33cde45ccaa844e378a 6, 19

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842 8, 31, 38, 42, 65, 70

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org. 12, 13, 14, 15, 16, 17, 18, 19, 45, 61, 62, 63, 64

[6] J. H. F. Trever Hastie, Robert J. Tibshirani, *The Elements of Statistical Learning: data mining, inference and prediction Second Edition*. New York: Springer Verlag, 2009. 12, 14, 16, 17, 35, 49, 67

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. 13, 19, 20, 21, 31, 38, 66

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf 13, 20, 21, 38

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385 13, 21, 31, 38, 41, 45, 66, 71

[10] E. J. H. Hamed Habibi Aghdam, *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detecton and Classification*. Cham: Springer, 2017. 14, 15, 17

[11] A. Geron, *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA : O'Reilly Media, 2017. 14, 16, 42, 67, 68

[12] R. Fletcher, *Practical Methods of Optimisation*, 2nd ed. New York, NY, USA: John Wiley & Sons, 1987. 16, 63, 64

[13] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *CoRR*, vol. abs/1411.1792, 2014. [Online]. Available: http://arxiv.org/abs/1411.1792 18, 19, 20, 21, 22, 48, 50

[14] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Trans. on Knowl. and Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2009.191 18

[15] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," *CoRR*, vol. abs/1311.2901, 2013. [Online]. Available: http://arxiv.org/abs/1311.2901 18, 21

[16] L. Shamir, T. Macura, N. Orlov, D. M. Eckley, and I. G. Goldberg, "Impressionism, expressionism, surrealism," *ACM Transactions on Applied Perception*, vol. 7, no. 2, pp. 1–17, feb 2010. [Online]. Available: https://doi.org/10.1145%2F1670671.1670672 20

[17] Y. Bar, N. Levy, and L. Wolf, "Classification of Artistic Styles Using Binarized Features Derived from a Deep Neural Network," in *Computer Vision - ECCV 2014 Workshops*, L. Agapito, M. M. Bronstein, and C. Rother, Eds. Cham: Springer International Publishing, 2015, pp. 71–84. 20

[18] B. Saleh and A. M. Elgammal, "Large-scale classification of fine-art paintings: Learning the right metric on the right feature," *CoRR*, vol. abs/1505.00855, 2015. [Online]. Available: http://arxiv.org/abs/1505.00855 20

[19] S. Karayev, A. Hertzmann, H. Winnemoeller, A. Agarwala, and T. Darrell, "Recognizing Image Style," *CoRR*, vol. abs/1311.3715, 2013. [Online]. Available: http://arxiv.org/abs/1311.3715 20

[20] W. R. Tan, C. S. Chan, H. E. Aguirre, and K. Tanaka, "Ceci n'est pas une pipe: A deep convolutional network for fine-art paintings classification," in *2016 IEEE International Conference on Image Processing, ICIP 2016, Phoenix, AZ, USA, September 25-28, 2016*, 2016, pp. 3703–3707. [Online]. Available: https://doi.org/10.1109/ICIP.2016.7533051 21, 23, 59

[21] A. Lecoutre, B. Negrevergne, and F. Yger, "Recognizing Art Style Automatically in Painting with Deep Learning," in *Proceedings of the Ninth Asian Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M.-L. Zhang and Y.-K. Noh, Eds., vol. 77. PMLR, 15–17 Nov 2017, pp. 327–342. [Online]. Available: http://proceedings.mlr.press/v77/lecoutre17a.html 21, 22, 23, 31, 32, 33, 44

[22] A. M. Elgammal, M. Mazzone, B. Liu, D. Kim, and M. Elhoseiny, "The Shape of Art History in the Eyes of the Machine," *CoRR*, vol. abs/1801.07729, 2018. [Online]. Available: http://arxiv.org/abs/1801.07729 22, 23, 28, 30

[23] C. Florea, R. G. Condorovici, C. Vertan, R. Boia, L. Florea, and R. Vrânceanu, "Pandora: Description of a Painting Database for Art Movement Recognition with Baselines and Perspectives," *CoRR*, vol. abs/1602.08855, 2016. [Online]. Available: http://arxiv.org/abs/1602.08855 23

[24] C. Sandoval Rodriguez, M. Lech, and E. Pirogova, "Classification of Style in Fine-Art Paintings Using Transfer Learning and Weighted Image Patches," 12 2018, pp. 1–7. 27, 28, 59

[25] A. D'Alleva, *How to Write Art History*. Laurence King Pub., 2006. [Online]. Available: https://books.google.co.uk/books?id=RoagE-LcrxwC 28, 30

[26] M. Hatt and C. Klonk, *Art History: A Critical Introduction to Its Methods*, ser. Art History: A Critical Introduction to Its Methods. Manchester University Press, 2006. [Online]. Available: https://books.google.co.uk/books?id=Q686y5HPDuQC 30

[27] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556 31, 38, 64, 65

[28] F. Chollet *et al.*, "Keras," https://keras.io, 2015. 41

[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *CoRR*, vol. abs/1512.00567, 2015. [Online]. Available: http://arxiv.org/abs/1512.00567 42, 65

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. 49

[31] R. D. S. Torres and A. X. Falcão, "Content-Based Image Retrieval: Theory and Applications," *Revista de Informática Teórica e Aplicada*, vol. 13, pp. 161–185. 60

[32] A. Bergamo, L. Torresani, and A. Fitzgibbon, "PiCoDes: Learning a Compact Code for Novel-Category Recognition," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., 2011, pp. 2088–2096. 60

[33] N. Qian, "On the momentum term in gradient descent learning algorithm," *Neural Networks*, 1999. 62

[34] E. H. J. Duchi and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimisation," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011. [Online]. Available: http://jmlr.org/papers/v12/duchi11a.html 62

[35] K. S. Geoffrey Hinton, Nitish Srivastava, "rmsprop: Divide the gradient by a running average of its recent magnitude." [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf 63

[36] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980 63

[37] M. Lin, Q. Chen, and S. Yan, "Network In Network," *CoRR*, vol. abs/1312.4400, 2013. [Online]. Available: http://arxiv.org/abs/1312.4400 65

[38] "Encyclopedia of Computational Neuroscience, Hebbian learning." 65