Painter by Numbers: Generalization for Unseen Artists using Siamese Networks
Final Report - Group 32

Nels Numan 4615107
Navin Raj Prabhu 4764722
Jose Soengas 4235762
Jakub Pietrak 4347293

1. Introduction

In this project we have evaluated approaches to the challenge proposed on Kaggle named *Painter by Numbers*[10]. The objective of this challenge is to distinguish whether two paintings were created by the same artist in the process of pairwise comparison. In a broad sense this could improve the identification of forgeries based on learned *artist style*. The dataset for the challenge is a collection of paintings from WikiArt.org[1]. After analyzing approaches taken by other competitors, we have identified a gap that could be explored: creating a network that not only learns the style from artists included in the provided dataset, but also is able to give a correct verdict for an artist that are not included in the training data. This creates an additional layer of complexity as the network has to facilitate ability to *generalize to unseen artists*. To tackle the problem at hand, a neural network architecture called the siamese network (Figure 1) was used.

After the analysis of approaches submitted by other competition participants, a baseline architecture 3.1 was chosen. Another approach that was explored is a siamese network composed of two symmetric branches, using equal weights of a pretrained Xception state-of-the-art CNN by Francois Chollet [2]. In the following sections these architectures are explored (3.2) and analyzed in the context of generalization performance on unseen data. Further improvements are proposed with regards to data set augmentation (3.3) and architecture modification. Altogether, three hypotheses are to be evaluated:

- *Hypothesis 1*: As a baseline model, how well can the CNN-based siamese network trained on the paintings dataset perform well in terms of generalisation error?
- *Hypothesis 2*: Will a siamese network trained on pre-trained features outperform the baseline CNN model?
- *Hypothesis 3*: Can further fine-tuning of the pre-trained features and feature representations give a better generalisation for unseen artist?

[1]`http://wikiart.org`

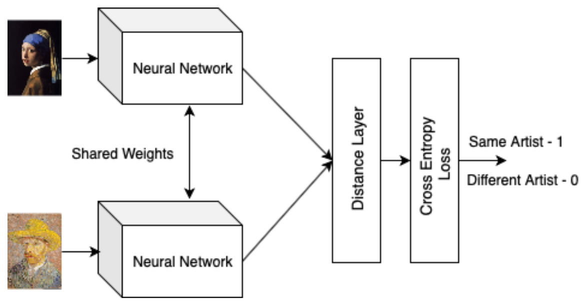

Figure 1. Siamese Neural Network Architecture

1.1. Related Work

Marco Venturelli [7] performed head pose regression for driver attention monitoring. The author used a siamese network to improve the training phase by learning more discriminative features. Since RGB images are sensitive to illumination and partial occlusions, depth-based approaches have been used. In our case however, texture and color information are the most informative parts, and therefore RGB images will be used. This will still allow for the siamese network to learn more discriminative features. In this paper, a novel loss function to improve the learning of the regression network was proposed. This loss function combines each of the two regression losses from the individual CNNs, and then finds the L2 distance between the prediction difference and the ground truth difference.

Transfer learning is the process in which the architecture and weights of a network, originally trained on one domain, are used to for another domain with distribution different to that of the original. This is achieved by initializing the training procedure with weights pertaining to the original network. Transfer learning usually provides an increase in performance for generalization on a new data set, even when only the last layer on the network is trained [5]. One may choose to retain all the layers, and in that way, retrain the entire network on the new data set. Another popular approach is to simply retrain the last layer of this new network on the new data set, and freeze the weights of all the other layers. This second method, is referred to as fine-tuning. This approach is useful, because as opposed to the first layers of the network, which appear not to be specific to a particular data set or task, the last ones do become more specific to the trained data set, and hence the most require a non-trivial change in weights Jason Yosinski and Lipson [5].

Sugimura [11] has proposed an architecture for solving the competition challenge. The author's architecture consists of training a siamese network with features from the Inception Resnet V2 model [12]. This approach embeds the the input images into 2048 vectors. Image pairs are then fed into a fully connected siamese network, where a selection of training pairs are evaluated. The pairs are split into similar and non-similar pairs. The author's results show a training accuracy of 96% with a validation set accuracy of 98% as well as precision, recall, and an F1-score of 97%. For this reason, we have chosen to subject this approach to extensive evaluating and to compare it to our baseline approach. The author does not specify his exact evaluation procedure, and the size of the test set is not clear.

2. Problem Statement

In this project we will be evaluating several existing approaches for the *Painter by Numbers* challenge. The main objective is to improve the existing performance on generalization to unseen data. This will firstly be attempted with a baseline Siamese CNN. The performance of this will then be compared to an architecture that incorporates a pre-trained CNN, which can embed our input images (for a Siamese net), baed on weights trained on ImageNet.

2.1. Dataset

The dataset that is used in this project was created for the *Painter by Numbers* [10] on Kaggle. This dataset contains 103,250 unique paintings by roughly 2300 different artist, of which 79,432 are in the training set and 23,818 in the test set. The majority of the images are sourced from WikiArt.org[2].

As can be seen in Figure 2, there is a significant amount of artists with 100 paintings or less in the full dataset. This could in turn cause an over-representation of certain artists in the training or testing phase. Furthermore, the majority of the paintings have the style impressionism, realism or romanticism (Figure 3).

2.1.1 Preprocessing

The dataset consists of a variety of different image shapes and resolutions. To make the images suitable for training and testing, we resize them to a size of 224×224 pixels. It can be argued that details such as brushstrokes and details of the paintings are discarded by this operation. However, due to performance limitations we scope this project

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

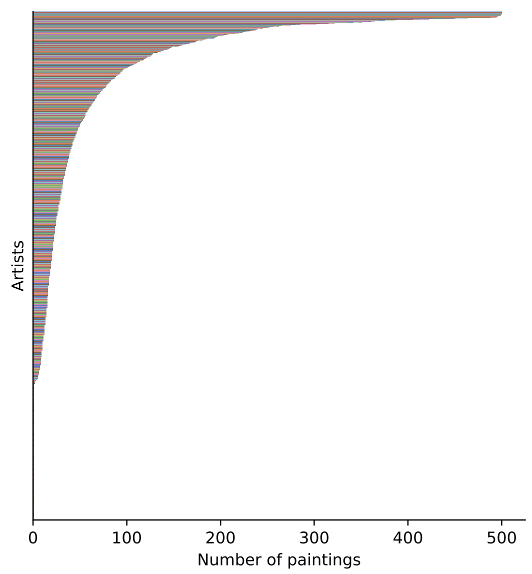

Figure 2. Number of paintings per artist in the full dataset, where individual artist labels are omitted

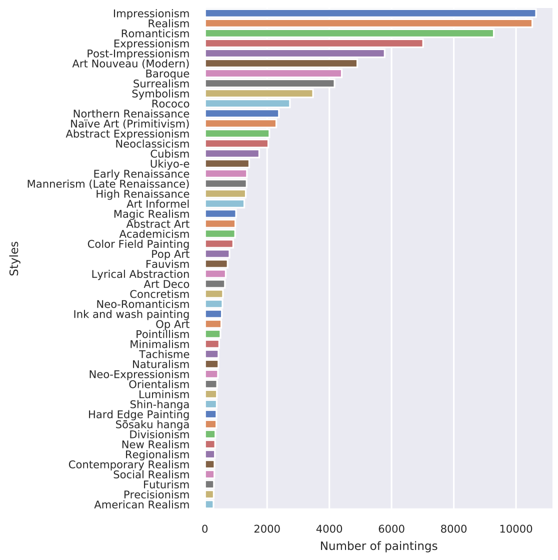

Figure 3. Number of paintings per style

to using small images. In the remainder of the project we will consider augmenting this data further in order to create more representative feature sets of each painting. One way of achieving this is by additionally taking 224×224 from the corners of the paintings, as described as a technique by [13]. In this project we will mainly be comparing pairs of images to evaluate whether they have been created by the same artist. This fact dramatically increases the size of our training set, as we are able to train on any unique combination of the 79,432 images in our training set. The pairs are found by firstly grouping all the paintings of a single artist and pairing all possible combinations within this group. In this way, we make similar pairs, so the siamese net can learn the discriminative features that related the same artist. For every similar pair, a dissimilar pair is also found by pairing one of the respective images in each similar pair, with a painting from a random artist. In this way, an equal balance of positive and negative examples are learned by the network.

2.2. Evaluation Metrics

To tackle the problem of *Painter by Numbers*, we explore *siamese networks* [1] which employ a structure to naturally rank similarity between training image pairs. Once this network has been tuned, we can capitalize on the powerful discriminative features to generalize not just to new data, but to entirely new classes from unknown distributions [**?**]. Keeping this in mind, in this project, we evaluate the performance of a network in terms of both *True Error* (performance on new data) and *generalisation error* (performance on new classes of data). Therefore, generalization will refer explicitly to the performance with respect to unseen data from a different distribution to that of the test data.

When training, the precision metric is used, quantifying the ratio of correctly predicted positive observations to the total predicted positive observations. Recall, will measure the ratio of correctly predicted positive observations to all observations in actual class. For completion, the F-1 score in provided.

3. Methodology

3.1. Baseline method: CNN-based siamese network

Convolution neural networks (CNNs) have been the state-of-the-art method for image classification. This is mainly due to the capability of CNNs to learn spatial features efficiently. But the performance of CNN on unseen classes are quite unknown and varies across domains of image data [1, 8**?**]. To explore the performance of CNNs on unseen classes in paintings, we framed Hypothesis 1 (Section 1). To test the hypothesis, we adopted a CNN architecture from the winner of the Kaggle competition [4]. The CNN was aimed at solving a multi-class painter classification problem and not a pairwise classification problem. With modifications on the adopted CNNs to fit it to the problem of pairwise siamese classification on the painting dataset, we aim to successfully validate Hypothesis 1 with respect to the *generalisation error* on unseen classes.

CNNs being an end-to-end method of learning, it takes more time to train the model on the pairwise comparison. The baseline **CNN-based siamese model** [4] in total had *13,351,872* trainable parameters and each epoch ran for approximately 12 hours. An illustration of the implemented baseline CNN model can be seen in the Figure 4. Since the baseline model was complex and required a large train time, the model was only trained for 10 epochs. The results of the model will be further explained in the Section 4.

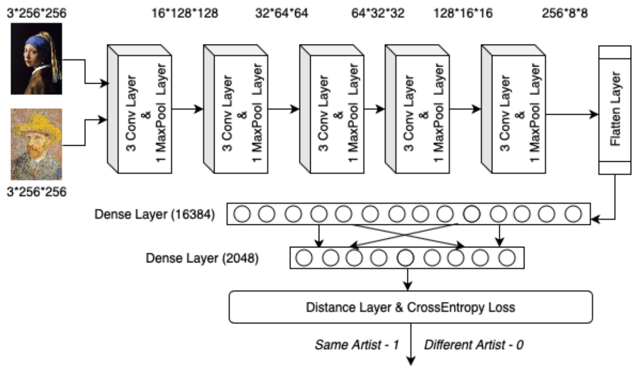
3*256*256
16*128*128
32*64*64
64*32*32
128*16*16
256*8*8
3 Conv Layer & 1 MaxPool Layer
3 Conv Layer & 1 MaxPool Layer
3 Conv Layer & 1 MaxPool Layer
3 Conv Layer & 1 MaxPool Layer
3 Conv Layer & 1 MaxPool Layer
Flatten Layer
3*256*256
Dense Layer (16384)
Dense Layer (2048)
Distance Layer & CrossEntropy Loss
Same Artist - 1
Different Artist - 0

Figure 4. An illustration of our CNN based Siamese Network

The CNN model achieved results of 64% for test dataset of seen artists, 56% for unseen artists and 59% on a mixed dataset after training the model for 10 epochs. While training this model it became clear that, the CNN model being an end-to-end learning model, it becomes complex and time consuming to train. Hence, we decided to adopt a pre-trained model. The task of *transfer learning* with the help of pre-trained model is explained in detail in the following section.

3.2. Transfer Learning

To investigate the added benefits of incorporating a pre-trained network into our global architecture, we reproduced the approach by Sugimura [11] based on the information provided. This approach consists of two stages. The first of which is the stage where features are generated based on a pre-trained network. To achieve this, every image in the dataset is run through a pre-trained network. In our case, we use the *InceptionV3* and *Xception* models. We will further elaborate on these two models in Section 3.2.

The features generated in stage one can in turn be fed to the second stage. The latter stage starts by the generation of a set of triples. These triples consist of the following:

- *Reference*: image that is used as a reference.
- *Different*: image that is from a different artist from the reference image.
- *Same*: image that is from the same artist as the reference image.

The above described triples are created by randomly picking an image of every artist. This image will serve as the reference image. To complete the triple, we randomly pick an image of the same artist and do the same once again, but for a different artist.

3.2.1 The *InceptionV3* & *Xception* model

Two state-of-the-art architectures are to be considered: InceptionV3 [12] and Xception [2]. This section briefly discusses the difference between the main components of architecture structure. To start with, it is important to state that the underlying hypothesis behind the use of both of those networks is the same: making the filter learning process easier and more efficient by explicitly factoring it into a series of operations that would independently look at cross-channel correlations and at spatial correlations [2]. The results have shown that this results in significant improvement on the image classification at decreased computational expense. The separation can be achieved by sequentially including a channel (RGB) cross-correlation 1×1 convolution and 'standard' spatial $n \times n$ convolution layers, where the order depends on the architecture choice. Inception implements this structure directly, while Xception reverts the order of action by first processing the input with spatial convolution performed independently over each channel of an input, followed by a point-wise convolution, i.e. a 1×1 convolution, projecting the channel's output by the depth-wise convolution onto a new channel space. In short, the Xception architecture is a linear stack of depth-wise separable convolution layers with residual connections. This makes the architecture easy to define and modify. For either, *InceptionV3* and *Xception*, when visualizing the output at different levels of depth in the network, is it noticeable that the layers that the deeper layers visualize more training data specific features, while the earlier layers tend to visualize general patterns like edges, texture, background etc [9].

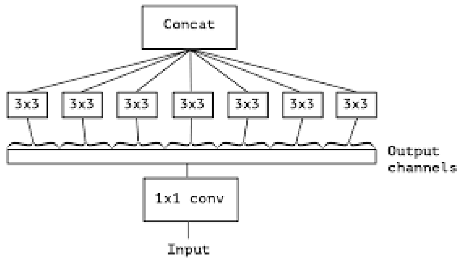

Figure 5. Depthwise convolution block - extreme version of Inception module with one spatial convolution per output channel of the 1x1 convolution.

Moreover, both models have the same number of trainable parameters, approximately 23 million. Both networks take an input of size $299 \times 299 \times 3$ and output the feature vector of size $8 \times 8 \times 2048$. The InceptionV3 model is a 42-layer deep canonical CNN model that contains three main Inception blocks using factorization, asymmetric convolution and asymmetric factorization respectively. The blocks are separated by grid reduction blocks. Xception contains 36 convolutional layers which are structured into 14 modules, all of which have linear residual connections around them, except for the first and last modules.

As an attempt to further improve our results, we **fine-tuned the Xception** model by adding a dense layer for our 1584 training classes. Firstly the top layer and subsequently the top block, namely block 14 (separable convolution, batch normalization and activation). In both cases, there were respectively, *3,245,616* and *7,994,416* trainable parameters.

3.2.2 Model visualisation

One of the goals of this paper is to build up on understanding of how a deep neural net learns the filters to classify abstract features (Like a painter's brush strokes or the colors preferred) of the data set consisting of paintings from various artists. To better understand how a siamese network performs on the task of categorizing an painting, a range of visual aids was evaluated. The most insightful of these is the heatmap representation that show the regions of the highest activation.

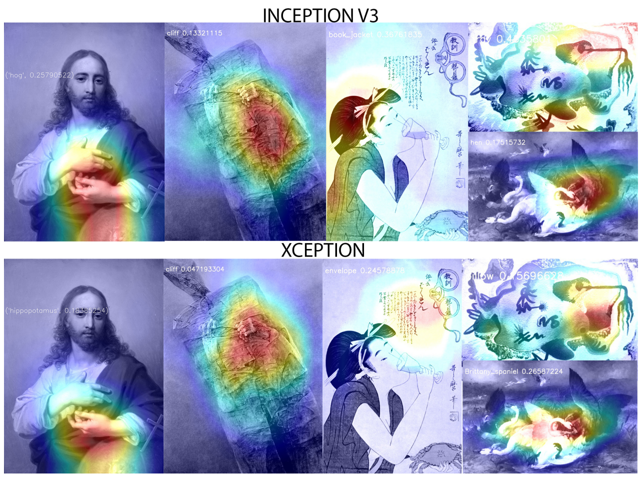

Figure 6. **Heatmaps** overlay over an input image represents the last convolution - classifier layer activation.

By plotting heatmaps superimposed on the input image we can visualize how the last convolutional layer focuses on the areas of the image corresponding to the highest class activation (Figure 6). Even though the predictions are not tailored to the challenge dataset, we can still observe what regions of the image the models focus on. In Figure 6, it can be seen that Inception and Xception often have a very different heatmap overlay. We can observe that Xception tends to focus on smaller and more detailed regions than Inception does. This could indicate that the Xception model is more fit to capture details in paintings, such as artist style and brush strokes, rather than focussing on structural elements which are mainly useful in object detection.

3.3. Concatenated feature vectors based on corners

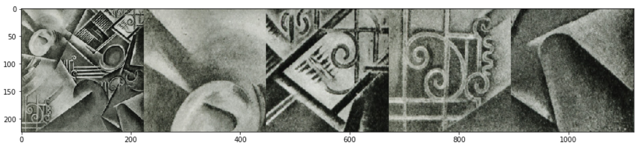

Figure 7. Concatenated images, each of size 224 × 224. From left to right, starting with the source image, followed by the corners in clock-wise order, starting with upper right.

To more closely capture the characteristics of each painting, we constructed a complex feature representation which concatenates the features of the original image of size 224 × 224 pixels with crops of each corner. An example of this can be seen in Figure 7.

This feature representation is obtained by firstly cropping each corner of the source image. We then calculate the individual feature vectors for the resized source image and each of the corners, which are each sized 224 × 224 pixels. This is done by using the same approach as what has been described in Section 3.2, but exclusively using Xception [2] as this model seemed to provide the best results. The resulting individual feature vectors are then concatenated to create a vector of size 10240. This feature vector was then used to train the aforementioned siamese model.

4. Results and Discussion

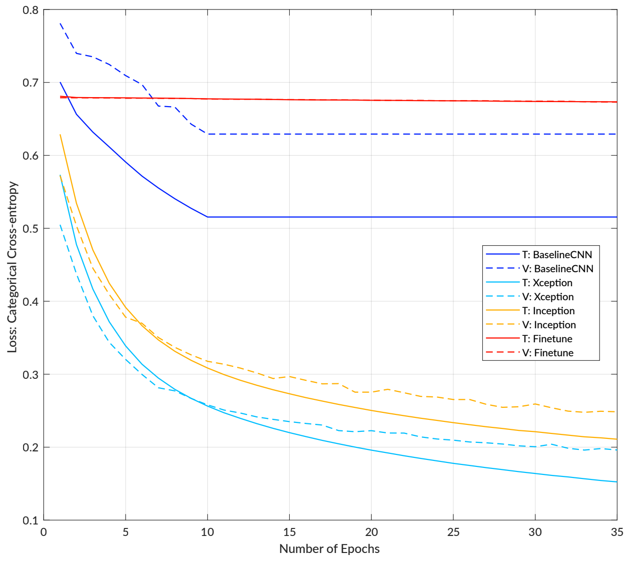

Figure 8. Learning curve for each approach

4.1. Baseline CNN

The baseline method was evaluated on the test set that was provided by the challenge. In order to be able to make conclusions about the generalization capabilities of the architecture, this test set was also split based on whether any paintings of the artist were included in the training set. Evaluating based on these three test sets, obtain the results in Table 1.

	Seen artists	Unseen artists	Mix
Baseline CNN**	0.64	0.56	0.59
InceptionV3	0.821	0.608	0.612
Xception	**0.888**	**0.647**	**0.672**
Fine-tuned	0.73	0.53	0.56
Corner features	0.512	0.501	0.502

Table 1. Results - Accuracy across the three test datasets for each of the model explained earlier. All models were run with one Tesla P100 GPU (3584 NVIDIA CUDA Cores with 16 GB of RAM). *****10** epoch runs, while others used **35** epochs.*

When looking at the results table, we can observe that the baseline model produces sub-optimal results. We can also observe that, for the baseline model, the validation and training curves in Figure 8 remain apart from each other during the training phase. This leads to believe that there is significant over-fitting while training a CNN based siamese model. Though the training was stopped after 10 epochs, It is also evident from the CNN's learning curves that the training/validation loss of the CNN converge gradually but rather slowly along the training epochs. So it is safe to also claim that the CNN's only and prime drawback was the complexity and without any constraints on time, the model may eventually converge to an optimal loss achieving a better accuracy.

4.2. Transfer Learning

There are a number of notable observations we can make about from the test results of the *Transfer Learning* model. Firstly, we can observe that the results are not as good as Sugimura [11] highlighted. This can have to do with a number of things. However, the most likely cause of this is that the evaluation method of Sugimura [11] was lacking in size of the test set. Lastly, we can observe that the *Xception* model performs significantly better than *InceptionV3* and for this reason we have chosen to fine-tune with the former. We can also observe that for both pretrained networks, both validation and training remain quite close to each other. This leads as to believe there is no significant over-fitting while training. From the Table - 1, we also see that the Xception model performs the best not only among the transfer learners, but also among other models used in this report. It is also evident that though the model performs well in the seen artists dataset with an accuracy of 88%, they drop in performance under unseen artists. Though this is a failure of the model, it is important to keep in mind that the unseen dataset is complex dataset and contains samples from completely different distribution (Paintings of Artists who are not used to train the model).

4.3. Fine-tuned Xception

An attempt was made to fine tune the pre-trained Xception CNN - previously trained on ImageNet - on our data set. This was done by performing global average pooling on the last layer of Xcpetion, and then adding a logistic layer with 1584 nodes, pertaining to the number of classes in our training data set. The logistic layer is activated using softmax. Firstly, the model is fine tuned on our training data set by only training the last layer. Then, the model is fine tuned on the top block of Xception, block 14. Fine tuning is done to try and learn features more inherent to our training data set. We did not get the results we were expecting when fine-tuning. In the end, the best validation loss was obtained by using stochastic gradient descent optimizer, and learning rate of 0.001, batch size of 128 with training validation split of 90-10. The validation loss over time for when fine-tuning the last layer is shown in 9.

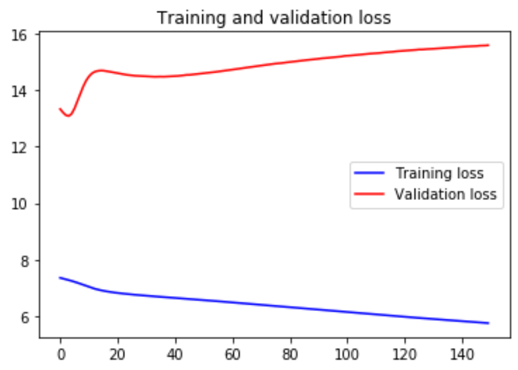

Figure 9. Learning curve of the Siamese net with fine-tuned model Xception.

In 9, one observes that there is a brief drop in validation loss at first, followed by an abrupt increase and then a steady increase. The reasons for this are not entirely clear. The model appears to be performing worse on the validation set whilst increasing training accuracy. Clearly, some over-fitting is taking place.

4.4. Concatenated Corner Features

The concatenated corner feature representation consists of the feature vectors of resized source image and each of its corners. The individual feature vectors having a size of 2048, this concatenation resulted in a final feature vector size of 10240. The large size of this vector has dramatically increased the complexity of the model and the time it takes to train this. Consequently, the results presented for this approach are notably low, as we have only been able to train the siamese model for two epochs.

5. Conclusion

Hypothesis 1: The baseline CNN model with $\sim$ 13 million trainable parameters takes a huge amount of time for loss convergence. We also note the **overfitting of the model** on the training dataset (Figure 8). Though the loss of the model converges moderately in the 10 epochs, it fetches a **sub-optimal generalization accuracy**.

Hypothesis 2: Siamese network trained on pre-trained features is less complex and produces better results (Table 1) than the baseline model. We also see there is **no overfitting** in the learning process (Figure 8). While the model performs **considerably well in the test set of seen artists**, it **fails to extrapolate** efficiently to unseen artists.

Hypothesis 3: Fine-tuning the Xception model **does not improve the performance** of the model. Reducing the feature vector to 1584 may have contributed to this. The concatenation of **corner features made the model too complex**, and dimensionality reduction or a smaller representation should be considered. Furthermore, the hyperparameters of the siamese model should be tweaked to accomodate the larger feature vectors.

6. Future Works

During our research on the *generalisation on unseen artists* using siamese networks, we have faced difficulties while extrapolating to unseen classes. While many models perform extremely well in pairwise comparison for seen classes, they fail for unseen classes. This is a common issue that can be observed with all the models we have evaluated in this report. While we still achieved moderate results, the question that remains not to be fully answered is: *what kind of feature representation helps achieve extrapolation to unseen classes?*. Hence, for future works in the same line of research, it would be worth analyzing feature representation techniques such as One Shot Learning [6] and Adversarial Data Augmentation [3]. Furthermore, it would be useful to spend more attention on complex feature representations such as the concatenated feature representation described in this report. It will be interesting to see their performance of these feature extraction techniques in pairwise comparison for unseen classes.

References

[1] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, pages 737–744, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. URL `http://dl.acm.org/citation.cfm?id=2987189.2987282`.

[2] Inc. Francois Chollet, Google. Xception: Deep learning with depthwise separable convolutions. 2017.

[3] He Huang, Changhu Wang, Philip S. Yu, and Chang-Dong Wang. Generative dual adversarial network for generalized zero-shot learning. *CoRR*, abs/1811.04857, 2018. URL `http://arxiv.org/abs/1811.04857`.

[4] Nejc Ilenic. Winning solution for the painter by numbers competition on kaggle. `https://github.com/inejc/painters`.

[5] Yoshua Bengio Jason Yosinski, Jeff Clune and Hod Lipson. How transferable are features in deep neural networks? 2014.

[6] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, 2015.

[7] Roberto Vezzani Rita Cucchiara Marco Venturelli, Guido Borghi. From depth data to head pose estimation: a siamese approach. 2017.

[8] James O' Neill. Siamese Capsule Networks. *arXiv e-prints*, art. arXiv:1805.07242, May 2018.

[9] Ankit Paliwal. Understanding your convolution network with visualizations. `https://towardsdatascience.com/understanding-your-convolution-network-with-visualizations-a4883441533b`.

[10] small yellow duck. https://www.kaggle.com/c/painter-by-numbers. `https://medium.com/@michaelsugimura/stylistic-fingerprints-of-artists-with-siamese-convolutional-neural-networks-2c0ab2188770`.

[11] Michael Sugimura. Stylistic fingerprints of artists with siamese convolutional neural networks. `https://medium.com/@michaelsugimura/stylistic-fingerprints-of-artists-with-siamese-convolutional-neural-networks-2c0ab2188770`.

[12] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[13] Luke Taylor and Geoff Nitschke. Improving deep learning using generic data augmentation. *arXiv preprint arXiv:1708.06020*, 2017.