DeepArt: Learning Artistic Style via Residual & Capsule Networks

{ Liam Eloie, Julia Gomes, Vasileios Papastefanopoulos, Eugene Valassakis}

Introduction

CNNs have been very successful in tasks such as object recognition, with many modern architectures able to outperform humans on ImageNet and other classification challenges. However, much less research has been applied to classification tasks pertaining to artwork. We believe that optimizing neural networks to learn artistic style would have widespread benefits for both artists and collectors alike. Our goal in this project is to:

1. Build an art classifier using the WikiArt dataset that can automatically label new pieces of art by "style" (e.g. cubism)
2. Investigate how architectures designed for object identification transfer over to the domain of artistic style classification

Data & Pre-Processing

Style Labels	
Class ID	**Type**
0	Art Nouveau Modern
1	Baroque
2	Cubism
3	Expressionism
4	Impressionism
5	Symbolism
6	Realism

Table 1. The different types of art styles and their corresponding class ID present in the reduced data set.

We used the WikiArt dataset, which is one of the largest publicly available art datasets consisting of around 100,000 paintings labelled by their artist, genre, and style. The paintings found in the WikiArt dataset [3] come from 26 different styles, spanning from Realism to Abstract Expressionism. We needed to reduce the number of data used to train the models. This was done by (1) Reducing the number of classes to be classified from 26 to 7, and (2) Choosing randomly 2000 paintings from each class, discarding the rest. Lastly, in order for the models to take these paintings as input, they must have the same dimension. To avoid the case where the cropping dimension is too small, leading to loss of important information, the smallest 10% of paintings were removed, then we cropped the remaining paintings to size.

CapsNet

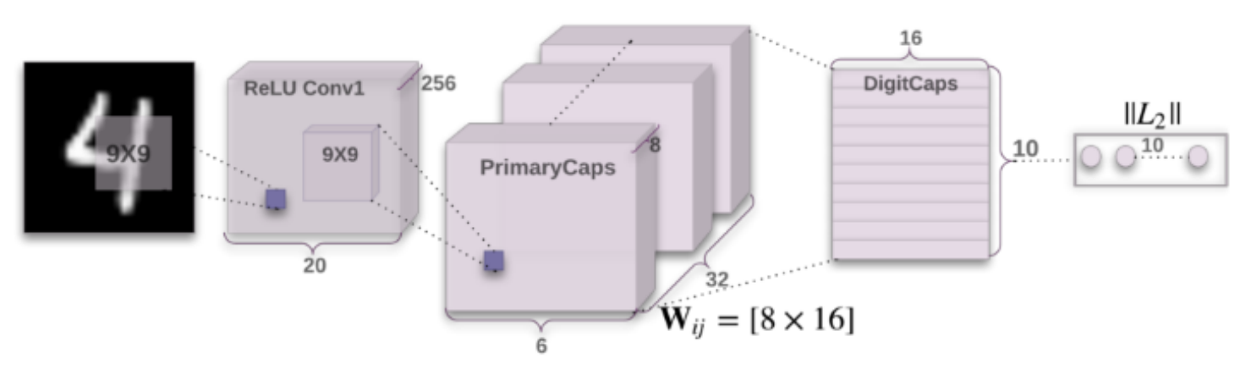


Figure 3. A simple Capsule Network Architecture (Sabour et al., 2017)

Capsule networks [2] use capsules to retain viewpoint invariant knowledge and spatial structure. These capsules perform complex internal computations on their inputs and then output a vector which encodes a variety of instantiation parameters related to a particular feature, such as position, orientation, scale, and texture. The length of the capsule output vector is squashed to be no more than one so that it can represent the probability that a feature is present, while the elements of the vector encode its properties. The network then uses a dynamic routing system called routing-by-agreement to send the vector to the appropriate parent in the next layer. For example, a capsule which observes noses will send its vector to the parent capsule looking for faces. This serves as an alternative to max-pooling. [2]
When we experimented with the CapsuleNet, we trained the model from scratch with a random initialisation. We did not use transfer learning because the CapsNet is fairly new and had only been trained on MNIST, which is very different from our dataset and has a resolution that is far too small for our purposes (28 by 28). Because of memory constraints on the Blaze machine, we applied a cropping dimension of (100,100) and used 4 crops per painting. We then trained the model over 10 epochs. The optimiser used was still Adam with a batch size of 16 (due to the limited memory resources).

Results and Discussion

Table 1: Resnet-50 Results

Frozen layers	Train Accuracy	Test Accuracy
None	0.6520	0.4340
Early layers	0.7410	0.4805
All except last	0.6200	0.5260

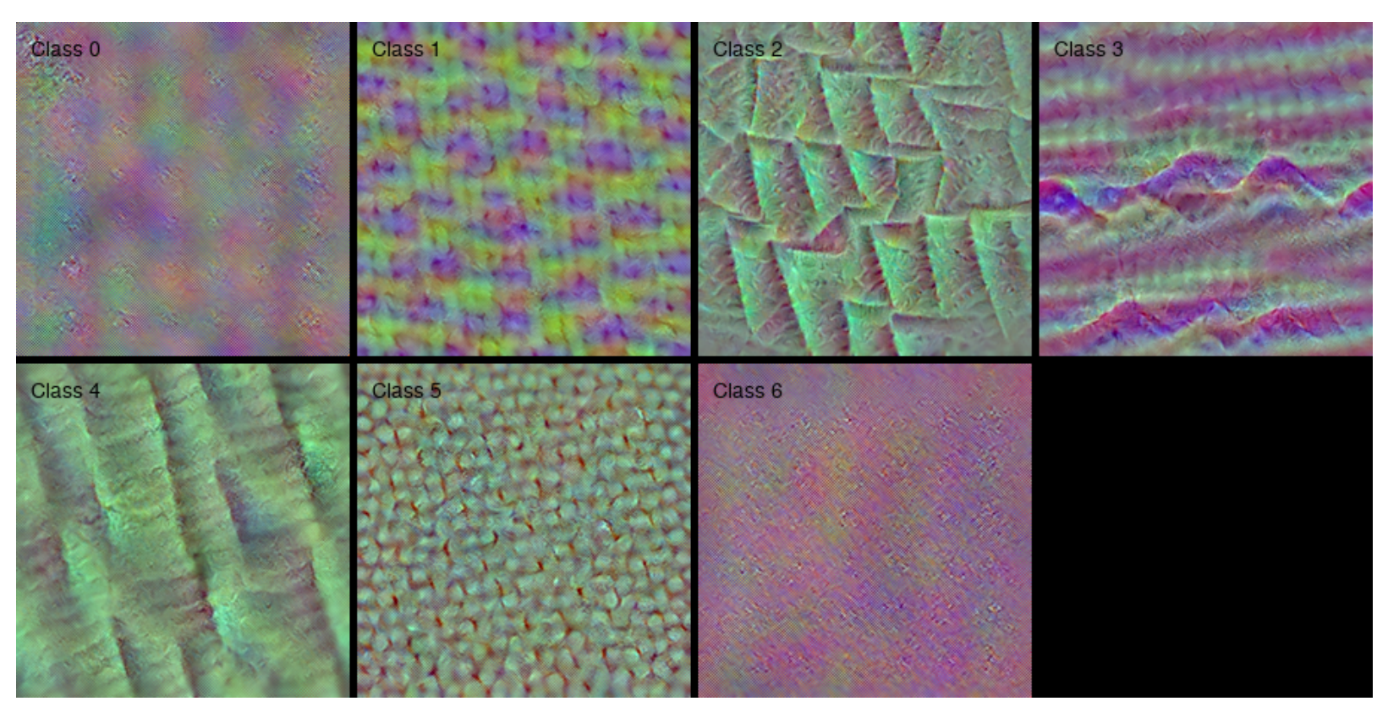
Table 3: CustomNet Results

	Train Accuracy	Test Accuracy
Standard	0.9810	0.5150
Using Dropout	0.9100	0.5175

Table 2: Capnset Results

Train Accuracy	Test Accuracy
0.1520	0.1680

Figure 1: Resnet-50 activation visualisations



Best Results: The above tables clearly indicate that the ResNet model trained on the ImageNet dataset is better at discriminating between our classes. In fact, the highest score is obtained while only fine-tuning the output layer of the network, leaving the pre-trained weights unchanged. Possible justifications for this behaviour are that our paintings dataset is not large or varied enough, or that we are still under-fitting the data when we train from scratch and that the model would benefit from longer training time. It is also worth noting that the model had a higher training accuracy when we re-trained the layers after activation 13 (leaving approximately $\frac{2}{3}$ of the network to re-train) than when we re-trained only the last layer, even though the testing accuracy was lower. This discrepancy indicates that we may have over-fit the model.

What to Improve? The CapsNet did not perform well. TheCapsNet is very slow to train and has an extremely large number of parameters, requiring both extensive training time and additional memory. We stopped training while the test set accuracy was on an upward trend. This seems to indicate that we are still in the under-determined case and that given more epochs to train, the CapsNet would have obtained a higher accuracy. We would like to test this theory using more resources in the future. Meanwhile, the CustomNet clearly over-fit our data, as the training accuracy is $\approx$ 98% and $\approx$ 91%, respectively, which is nearly double the testing accuracy. The test accuracy on CustomNet is not higher than the test accuracy for the ResNet model in which we trained all except the last layer. This suggests that concatenating the capsule layers to the ResNet output does not improve performance.

Interesting Features: We can observe the visualization of the last layer activations for each of our style categories. The most illustrative example is the one of Experiment 2, which is shown above. It is clear that our network learned to recognize some of the styles better than others. For instance, in class 2 we can see sharp, geometrical edges cutting off each other at various angles creating an mesh of inter-tangled geometrical shapes. This is quite reminiscent of the style of Cubism which it represents!

ResNet-50

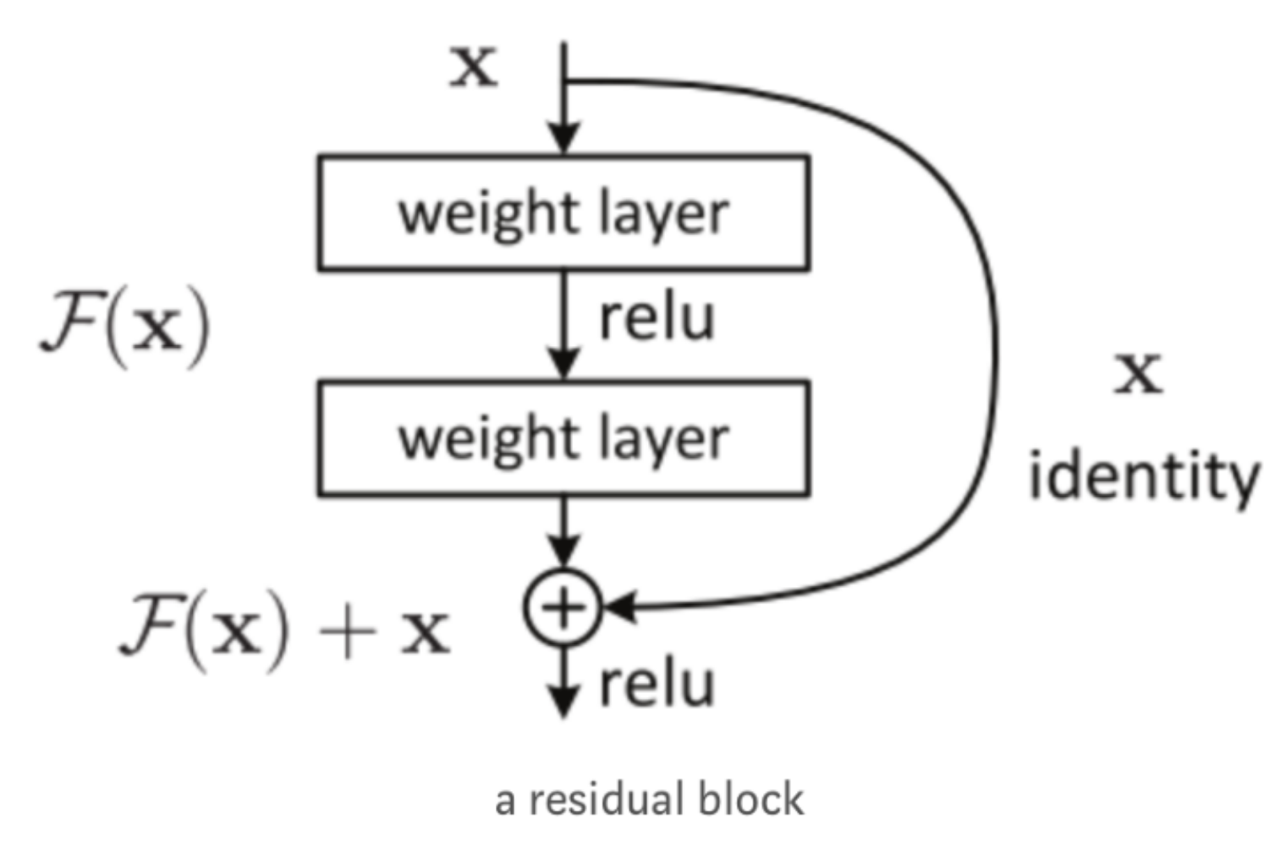


Figure 1. An illustration of the residual block (Fung)

We chose the ResNet [1] because it has high accuracy for a relatively small number of parameters. In addition, ResNet avoids the vanishing gradient problem by introducing an identity short-cut connection, illustrated above. This allows the ResNet to use identity blocks to imitate a shallower model, so that adding layers to the ResNet will not degrade performance.

We experimented with transfer learning on the ResNet-50 network. Transfer learning works by taking a neural net trained on another dataset, then "freezing" some initial layers (or all but the last layer) and training the rest of the network. The idea is that if two datasets are similar, the features learned from one dataset will be relevant to the other dataset. We wanted to test if the features learned from an object classification task would be relevant to style classification. Specifically, we used the pre-trained weights on the ImageNet dataset, and then fine-tuned them using our WikiArt dataset.

1. **Experiment 1**: We fine-tune all layers of the ResNet using our WikiArt dataset
2. **Experiment 2**: We freeze all layers up to and including the first 13 layers of 49, then train the remaining 36
3. **Experiment 3**: We only fine-tune the final classification layer of the network

CustomNet

We noticed that the first block of the CapsNet was composed of a regular CNN, and hence thought it would be interesting to replace this by the ResNet network. The rationale behind this is that it might be more effective to use the representations of the ResNet (trained on the massive ImageNet dataset), rather than learning representations from scratch through a standard convolutional layer (and our much smaller training set). As such, we created a custom network, which we named CustomNet, in the following way:

1. First, we used the convolutions of the ResNet (stripping away the ResNet output) and loaded the ImageNet weights.
2. We then froze this part of the network so that these weights would be unaffected by training.
3. Finally, we routed the output of this configuration through the capsule layers of the CapsNet, which produced the final seven class outputs.

In order to respect consistency in layer-by-layer dimensionality, we modified the kernel size of the capsule layer from 9 (value used in the CapsNet) to 3. We experimented by adding a dropout layer with dropout probability 0.5. We train the CustomNet using input dimension (224,224), 10 epochs, batch size 32, the Adam optimiser, and the margin loss function.

References

[1] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *arXiv preprint arXiv:1512.03385v1* (2015).

[2] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. "Dynamic routing between capsules". In: *Advances in Neural Information Processing Systems*. 2017, pp. 3859–3869.

[3] WikiArt. *WikiPaintings*. https://www.wikiart.org, Accessed: Dec. 2017.