Multi-label classification of art using deep neural networks
Final project, May 2019

Ksenia Sokolova
CS department, Princeton University
sokolova@princeton.edu

Abstract

Digitizing large museum collection is a labor intensive task, that would benefit from automatic or semi-automatic labeling of the artworks. This project uses the digitized and annotated collection of art by the Metropolitan Museum of Art to train a deep convolutional neural network to predict 1103 labels. The best model achieves the mean F2-score of 0.533 and AUC of 0.94 on the test set. Additionally, the first 8 layers of trained model can be used as a content aware crop tool for object photography.

1 Introduction

The largest museums typically display about 5% of their collection at any time [1]. Tucked somewhere in the storage are works of Monet, Picasso, Schiele and many others. But while museums might not have the physical space to display all the artworks, there is an increased effort of digitizing the collections [2][3][4]. It is a costly task, that includes not only taking photographs of the art, but also labeling images during post-processing. As such, an automatic method of labeling artworks would be cost-reducing improvement for the digitizing pipeline.

Dataset used in this project is a digitized and annotated collection of art by the Metropolitan Museum of Art [5]. It was released as part of the competition for the CVPR'19 workshop on fine-grained visual categorization [6]. The dataset includes photos of the artwork and annotations. The images are normalized such that the shorter dimension is 300 pixels. There are 1103 attributes, out of which 398 are culture tags, and 705 are objects present. Each image has at least one label assigned to it.

2 Related work

The task of labeling artworks falls into the category of image classification in general, and specifically involves fine-grained classification. Fine grained classification aims to recognize objects that have a high degree of similarity, and thus the models have to rely of marginal differences between images [7]. At the same time the model needs to be general enough to accommodate for the different types of art objects.

In computer vision field, deep convolutional neural networks (CNNs) rank among the most popular and well performing models. They are considered one of the most representative classes of models in deep learning The modern popularity of CNNs in computer vision started in 2012, when AlexNet [8], a deep convolutional neural network (CNN), significantly outperformed other models in the ImageNet classification competition. There are currently about 5 CNNs so commonly used in the field, they are available in the pre-trained state from most major deep learning libraries [9] [10].

It was shown that when the number of images per class is low, transfer learning should be used [11]. Transfer learning is a process of using a model with pre-trained weights, instead of a random initialization. VGG16 [12] is the model that is used in this project. VGG16 stands for "Visual Geometry Group" (University of Oxford) model with 16 layers. It was published in 2014, and won ImageNet competition of that year.

3 Methods

3.1 Dataset

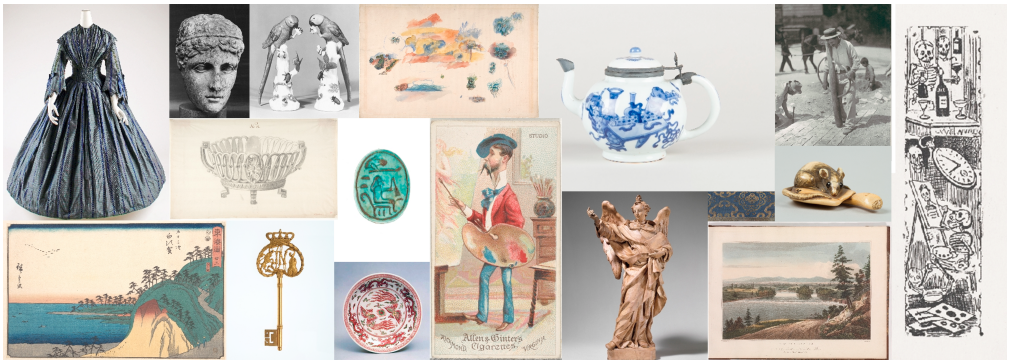

Figure 1: *Examples of images from the dataset*

There are 109,237 images in total in the dataset, all normalized to have 300 px short dimension. See some sample images in Fig.1. Before training dataset was randomly split into train (80%) and test (20%) datasets using *sklearn*[13] train/test splitting function.

Note that the camera position is not known and varies depending on the object. The background color for the objects also varies. To further complicate the matters, annotations are created without the verification step - only one person is responsible for the labeling. The annotators could also provide a textual commentary. Such annotation practice introduces noise.

3.2 Convolutional neural nets

Convolutional Neural networks are designed to process data that comes in a form of multiple arrays and where spatial relative positions of the pixels play an important role. The typical CNN for classification consists of 2 blocks. The first block includes convolutional layers and max pooling, while the second one - fully connected layers [14].

Convolutional layers are based on the process of convolution, which, roughly speaking, applies a matrix to the image by sweeping the filter matrix over the image, with element-wise multiplication and summation. Mathematically, given the image I and filter H, the convolution image J is [15]

$$J(r,c) = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} H(u,v)I(r-u, c-v)$$

Units in the convolutional layer are organized into feature maps, and every unit is connected to the patches of feature maps in the previous layer through a set of weights/filters [14]. This means that the first layers of the convolutional net extract local motifs, and these motefs itself are invariant to the location, or in other words the motif search is applied to the whole image.

Max pooling is a down-sampling method, that extracts the maximum of the matrix and thus reduces the spatial dimensionality of the image. The role of pooling features is to merge semantically similar features into one [14].

VGG16: as stated previously, VGG16 [12] is a convolutional neural network comprised of 16 layers (See Fig. 2). In this project, only the 2nd block of the network was changed, namely the fully connected layers. The shape of the output was changed from 1000 (original) to 1103 to match the number of classes. The intermediate fully connected layer was varied from 5000 to 6000 neurons. The input image size is 224px by 224 px. All the images need to be normalized using the mean per-channel array of $[0.485, 0.456, 0.406]$ and per-channel standard deviation of $[0.229, 0.224, 0.225]$. The stochastic gradient descent was used.

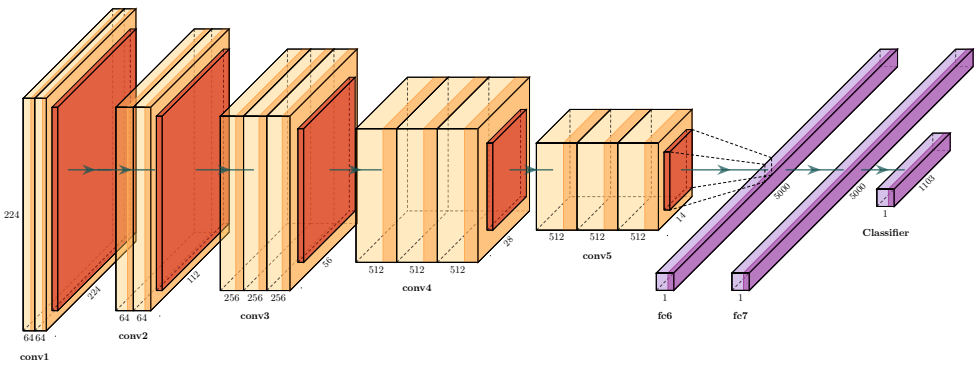

Figure 2: *VGG16 architecture used in this project*

Batch-normalization: to reduce the dependency of model training on the starting learning rate and to improve regularization, batch normalization can be used [16]. It is a process of normalizing activations in the intermediate layers of deep neural networks performed across mini-batches.

3.3 Data Augmentation

Data augmentation is a popular technique that allows to artificially increase the training dataset size and avoid overfitting. Traditional methods involve computationally simple changes, for example flipping the image matrix, to obtain new images. In this project, a set of transforms, implemented by PyTorch [9], are used: random horizontal flip with probability 0.5, random rotation ± 7 degrees and random gray scale with probability 0.1. See example of random transforms in Fig.3.

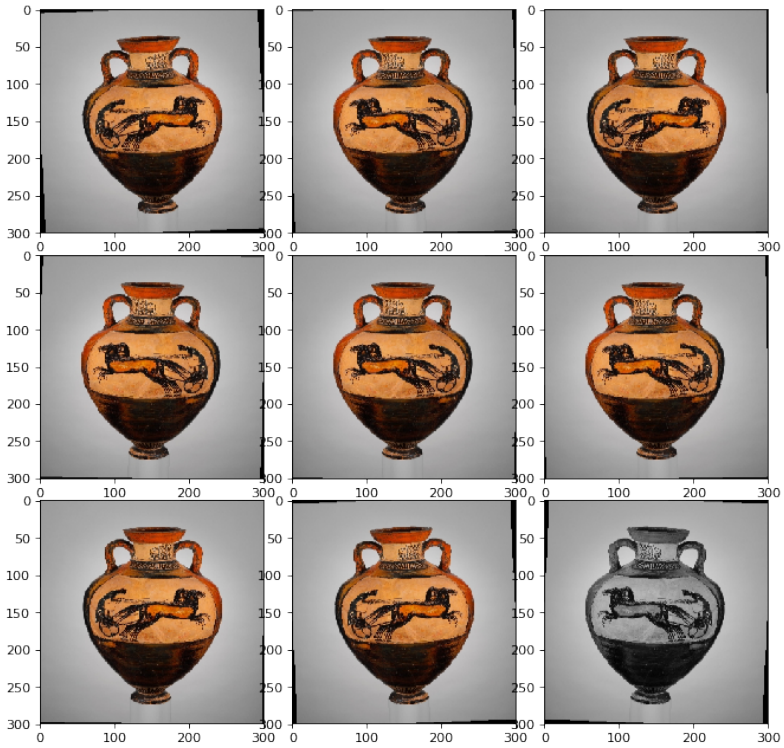

Figure 3: *Example of random tranformations: the same image drawn multiple times from the dataset*

3.4 Selecting learning rate

Finding the best learning rate for the model is a significant step, as some say it is the most important hyper-parameter. A simple way to perform a search would be to run training many times with different starting settings. However, this is time consuming. [17] introduced an "LR range test", a method to estimate the best learning rate. The learning rate is increased linearly between two boundaries, and accuracy is recorded. The optimum learning rate is then roughly between the point where accuracy starts increasing and when it slows down. The paper discusses this in application to the cyclic learning rule. In this project, a simpler model of step learning rate (StepLR) change is used: every N epochs, the learning rate is decreased by some fraction. A publicly available implementation [18] of lr-finder was used.

3.5 Assessing quality of the results

F2-score: This score is a subtype of the F-beta score, which is a weighted harmonic mean of precision and recall. The optimal value is 1, and the worst value is 0. Mathematically,

$$F_\beta = (1+\beta^2)\frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$$

where

$$\text{precision} = \frac{TP}{TP+FP}$$

$$\text{recall} = \frac{TP}{TP+FN}$$

and TP are true-positives, FP are false positives, FN are false negatives.

The F2-score is calculated by setting $\beta = 2$. Here, F2-score is applicable because the cost of false negative - not adding a label to the image, should be higher than false positive. This is due to the nature of the problem - the goal is to label images so that the database is easily searchable. Image without labels will not show up, while extra non-applicable entries could be easily removed later by the users request. At the same time, search should not be full of extra images, and therefore setting β higher than 2 is not of interest.

Focal Loss: During training, the goal of the model is to minimize loss and the loss function used has a great effect on the performance. A team at Facebook AI introduced a Focal Loss measure [19] to deal with class imbalance, that adds a weight term in front of the usual cross entropy loss to focus on hard examples.

Take each individual label. Let y be the ground truth label, $y \in \{\pm 1\}$ and $p \in [0,1]$ be model's estimated probability for the class with label 1. Define p_t as:

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{if } y \neq 1 \end{cases}$$

The cross-entropy loss becomes

$$CE(p_t) = -log(p_t)$$

Focal loss model then adds a modulating factor $(1-p_t)^\gamma$ to the cross-entropy loss:

$$FL(p_t) = -(1-p_t)^\gamma log(p_t)$$

In this project, the value of is $\gamma = 2$. [20] implementation used.

ROC and AUC: Receiver Operating Characteristics curve (ROC) is a graph showing the performance of the model at different threshold levels. It is a plot of the false positive rate (FPR) vs true positive rate (TPR), where $TPR = \frac{TP}{TP+FN}$ and $FPR = \frac{FP}{FP+TN}$. AUC is calculated as an area under the ROC curve, and it provides an aggregate measure of performance across all classification thresholds [21]. The best value possible is 1.

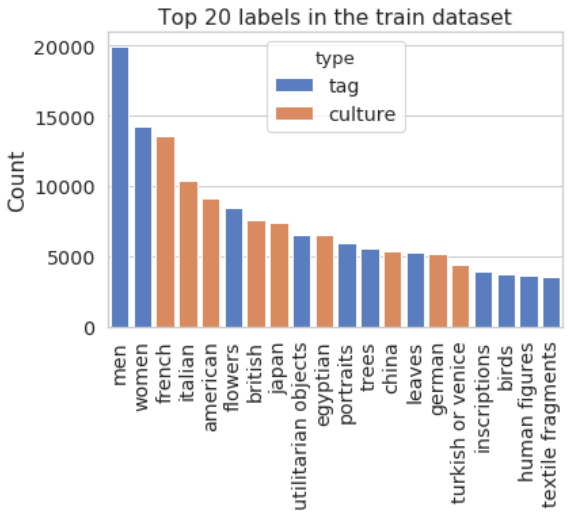

Figure 4: *Top 20 labels by the number of images associated with it in the whole dataset*

	Model	Avg. AUC	F2 score	Focal Loss	Batch size	Learning rate	StepLR
1	VGG16	0.932	**0.533**	3.082	10	0.001	0.1/10
2	VGG16_bn	0.928	0.51	3.052	128	0.01	0.1/10
3	VGG16_bn	**0.94**	**0.533**	**2.91**	128	0.01	0.1/15
4	VGG16_bn*	**0.94**	0.53	2.92	128	0.01	0.1/15

Table 1: *Performance summary of the models trained. The * denotes a model with 6000 neurons in the fully connected layer. _bn denotes models that use batch-normalization*

3.6 Auto-cropping

To perform automatic cropping, the first 8 layers were used. For VGG16 without batch normalization, each feature layer was additionally normalized with respect to itself, and layers with low standard deviation were removed. The remaining layers were summed up. Nearest interpolation is done to scale feature maps up, since the size was smaller than than of the original image. The peak localization was performed using *scipy* [22] and *scikit*[13]. The location of the peaks was used to find the crop lines.

4 Results

The dataset is unevenly split between different labels. The most common tag-type labels are *'man'* and *'women'* and those appear in the whole dataset 19970 and 14281 times respectively. The top 20 labels can be seen in the Fig 4. At the same time, there are 15 labels, for example *culture: zoroastrian*, that appear in the whole dataset only once. The mean number of images per label is 314, and median is 57.

The images were normalized to have the short side be 300px. The mean length of the other side is 443px, and the median is 402px. The distribution of the longer length can be seen in Supp.Fig.10. From the plot we see that there are very long images, with the longer side greater than 2000px. As a matter of fact, there are in total 213 images with longer dimension greater than that.

Four different models were trained. The summary of the performance of the models can be seen in Table 1. Note that "_bn" denotes models that use batch normalization, and the last model (4) had 6000 neurons in the fully connected layer, while all other ones had 5000. Increasing the number of neurons while keeping all of the parameters fixed did not significantly alter performance of the model. Adding batch normalization and keeping the learning rate high for more epochs produced the best performing model (3), based on all 3 measures. Note that model 1 failed to converge with starting learning rate greater than 0.001.

The test Focal Loss calculated after every epoch per model can be seen in Fig. 5. From the graph it is clear that models 3 and 4 had a very similar test error change. Models 1 and 2 achieved similar best focal loss, however Model 1 started to exhibit signs of over-fitting after Epoch 10, since the training error kept decreasing, but the test error started to grow. Comparison of the train and test scores per model during training is available in Supp.Fig. 11.

The F2 score is dependent on the threshold, and as such the measure reported is the maximum obtained by trying different values. See example plot of the F2 score vs threshold in Supp.Fig. 12.

The ROC curves per class for the best performing model 3 can be seen in Fig. 6. Additionally, a more detailed overview of the performance on the top 20 most common classes can be seen in Table 2. Among labels with more than 20 examples in the test set, the best AUC was associated with *tag: netsuke* (miniature sculptures with practical purpose, such as shown on Fig 8), and the worst AUC score was associated with *tag: peacocks* (See Tables 3 and 4).

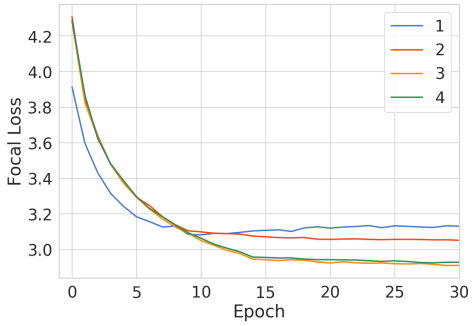

Figure 5: *Test error vs epoch for the 4 models used. Observe that, after epoch 10, test error starts to increase for model 1. Models 3 and 4 exhibit similar trends.*

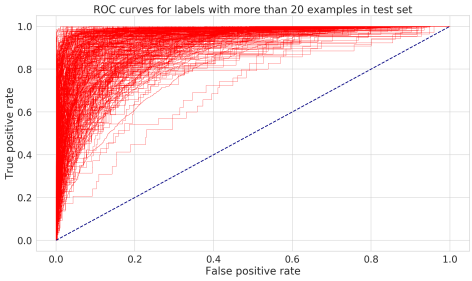

Figure 6: *ROC curve for labels with more than 20 examples in the test set.*

Label	# in Test	AUC
tag::men	3991	0.911401
tag::women	2831	0.916577
culture::french	2705	0.875143
culture::italian	2138	0.923297
culture::american	1783	0.969055
tag::flowers	1684	0.93337
culture::british	1543	0.904896
culture::japan	1472	0.980762
culture::egyptian	1323	0.980658
tag::utilitarian objects	1310	0.904554

Table 2: *AUC scores for the most common labels in the test set*

Label	# in Test	AUC
tag::peacocks	35	0.719097
tag::emblems	29	0.726064
tag::swans	22	0.789386
tag::grapes	38	0.799219
tag::animals	571	0.801284

Table 3: *Worst performing labels with more than 20 examples in test, model 3*

Label	# in Test	AUC
tag::netsuke	111	0.999371
tag::cuneiform	70	0.999342
tag::actresses	286	0.998523
tag::architectural elements	44	0.99852
tag::sword guards	71	0.998344

Table 4: *Best performing labels with more than 20 examples in test, model 3*

Careful analysis of the input images and predictions reveals four different types of mislabelling issues: absence of the details tag label, absence of the more general tag label, labels requiring

outside knowledge and duplicate labels. For example, as can be seen from Fig 7, the image of textile on the left is missing the true label *leaves*. The image of pants is missing label *clothing and accessories* and image of the ring is missing label *jewelry*. The plaque of a man's profile has a label *doctor*, which can be deduced only if one reads the inscription 'Pierre Potain' and knows that he was a cardiologist. Finally, there are two classes *textile* and *textile fragments*, that are used interchangeably in the dataset (there is also a tag *Christ* and a separate tag *Jesus*).

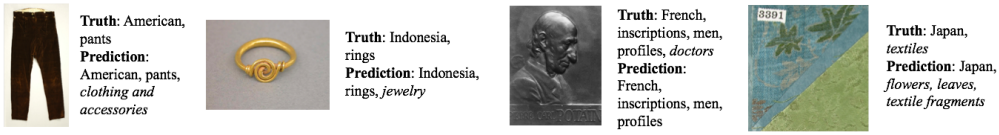

Figure 7: *Example of errors introduced by labeling inconsistencies*

The first 8 intermediate layers of the convolution can be used both as a content aware crop and to get insight into the model behavior. Note that if VGG16 model without batch normalization is used, additional normalization is needed to achieve this results. Each feature map needs to be normalized with respect to itself, maps with low standard deviation removed and the remaining ones are summed together. After peak extraction, these features are used as a crop guidelines. See Fig. 8 for a detailed pipeline. The process works well for the objects with a uniform background.

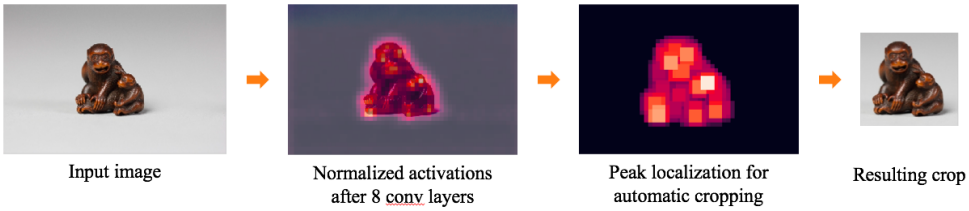

Figure 8: *Example of auto-cropping pipeline. The first 8 convolutional layers are used to extract the activations, then the feature maps are passed through the peak localization function and the image is cut based on the location of the peaks.*

As a diagnostics tool, the feature activations can be used to gain insight into the areas of 'focus' and changes after fine-tuning. For example, in Fig 9, the original image and crops produced from pre-trained and fine-tuned models are shown. Specifically, the pre-trained model used is VGG16, while the fine-tuned model is # 1 in the Table 1. The process of cropping is the same for both. It is clear that the activation of the model have changed to include more background.

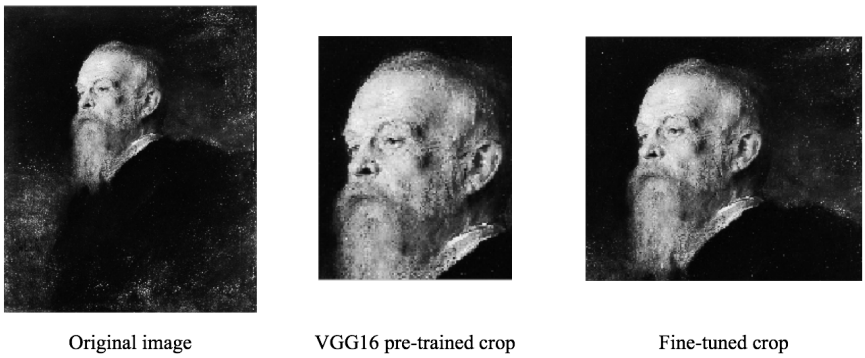

Figure 9: *Example of changes in automatic cropping from pre-trained to fine-tuned VGG16. Note the change in the amound of the background included after fine-tuning the model.*

5 Discussion and Conclusion

The most popular labels in the dataset are interesting not only as a dataset information, but also as an insight into the type of art that is stored and used by the Met. The three most common cultures represented are French, Italian and American, closely followed by British and Japanese. That is not surprising, granted the enormous impact the listed cultures had on the art development, and the fact that the Met is located in America. However, a more detailed examination of the cultural representation in the museum might be interesting.

The imbalance in the dataset in both the number of images per label and the size of the images can pose certain problems. However, the narrow images comprise only a small part of the dataset overall, and thus should not have a great impact on the model performance. The imbalanced labels problem should be alleviated by using the Focal Loss.

Overall, all four models achieved good performance. The AUC score was above 0.9 for all the models used. Interestingly, there is no strong dependency between the F2 score and focal loss: the best F2 score was achieved by models 1 and 3. While model 3 had the lowest Focal Loss among the three, the model # 1 had the highest. Changing the step of the learning rate had a drastic effect on the performance of the models, where model 3 outperformed model 2 for all the measures, with the only change being the step size. As such, an interesting future work could be to use more advanced methods of the learning rate change. That could involve exploring more versions of the step-change, or trying cyclical and other learning rate change patterns.

Increasing the number of neurons from 5000 to 6000 did not have any effect on the performance of the model. That is an interesting fact, as it could either mean that the model did not converge properly, so that for example some of the neurons were not used optimally, or that the model is saturated with the 5000 neurons and the bottleneck is in the previous layers. Nevertheless, increasing the number of neurons makes the model larger, complicates convergence and in the absence of improvement is not justified.

The ROC-curves trends show generally an arguably even performance across all labels. The worst AUC score for examples with more than 20 images in test set was 0.72, which is still decent, and the best was 0.999. Each label offered unique difficulties, and it seems that animals in particular were complicated. Out of top-5 worst AUC scores, 3 are in that category. At the same time, recognizing objects and their properties was among the easiest (netsuke, cuneiform, architectural elements and sword guards are among the top performing labels). It is interesting that *culture* labels did not make either top-5 or bottom-5 labels. For future work it would be beneficial to take a closer look at the worst-performing labels to try to deduce whether the performance is due to the true complexity of the images, or whether there are some labeling issues.

Content-aware crop using the intermediate layers is interesting in its own right - and can be applied as a batch processing tool for the object images to have consistent crop styles. The changes in the activations after fine-tuning, as evident from Fig 9 provide an insight into the way the CNN treats recognizing art. Originally trained on the ImageNet, the main area of focus is the face. However, after fine-tuning, the focus expands to include the background. It can be argued that the change is due to the fact that the artistic style now matters, and the background provides additional information such as stroke styles.

Finally, the problem of discrepancies in the labeling is arguably one of the most promising areas of improvement. As is, the labels provide a very noisy and unreliable source for training. Each of the problems, as outlined in Results section, can be treated separately. The absence of the generalized class is the easiest, as it can be added using either some computational method, or sifting manually through classes and creating a tree-structure of the labels. Similarly, the problem of synonyms present is easy, as those labels can be merged together upon manual examination. The last two problems: the need of the external information and absence of some tags is more complicated. To correct those problems, most likely an additional re-labeling would be required, and the discrepancies between the labeling versions would need to be accounted for.

In conclusion, in this project 4 different deep convolutional neural networks were trained. The performance of the models was compared, and the best model was selected. The detailed performance across all the labels was further analyzed to deduce possible pitfalls and future improvement options. Additional uses of the trained model, such as automatic cropping of the objects images, was shown.

References

[1] C. Groskopf, "Museums are keeping a ton of the worlds most famous art locked away in storage." https://qz.com/583354/why-is-so-much-of-the-worlds-great-art-in-storage/. Accessed: April 20, 2019.

[2] "Digitization of smithsonian collections." https://www.si.edu/newsdesk/factsheets/digitization-smithsonian-collections. Accessed: April 20, 2019.

[3] "About the digitization initiative at the metropolitan museum of art libraries." https://libmma.contentdm.oclc.org/digital/about/. Accessed: April 20, 2019.

[4] "MoMA: archives." https://www.moma.org/research-and-learning/archives/. Accessed: April 20, 2019.

[5] "iMet collection 2019: dataset." https://www.kaggle.com/c/imet-2019-fgvc6/data. Accessed: April 20, 2019.

[6] "The sixth workshop on fine-grained visual categorization." https://sites.google.com/view/fgvc6/home. Accessed: April 20, 2019.

[7] H. Xu, G. Qi, J. Li, M. Wang, K. Xu, and H. Gao, "Fine-grained image classification by visual-semantic embedding," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 1043–1049, International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[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.), pp. 1097–1105, Curran Associates, Inc., 2012.

[9] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[10] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[11] L. Torrey, "Chapter 11 transfer learning," 2009.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[13] 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.

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015.

[15] C. Tomasi, "Image correlation, convolution and filtering," 2017.

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[17] L. N. Smith, "Cyclical learning rates for training neural networks," *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar 2017.

[18] D. Silva, "PyTorch learning rate finder."

[19] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[20] bestfitting, "A CNN classifier and a Metric Learning model,1st Place Solution."

[21] "Classification: ROC curve and AUC," *Google Machine Learning: Machine Learning Crash Course*.

[22] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online; accessed ¡today¿].

6 Supplemetary materials

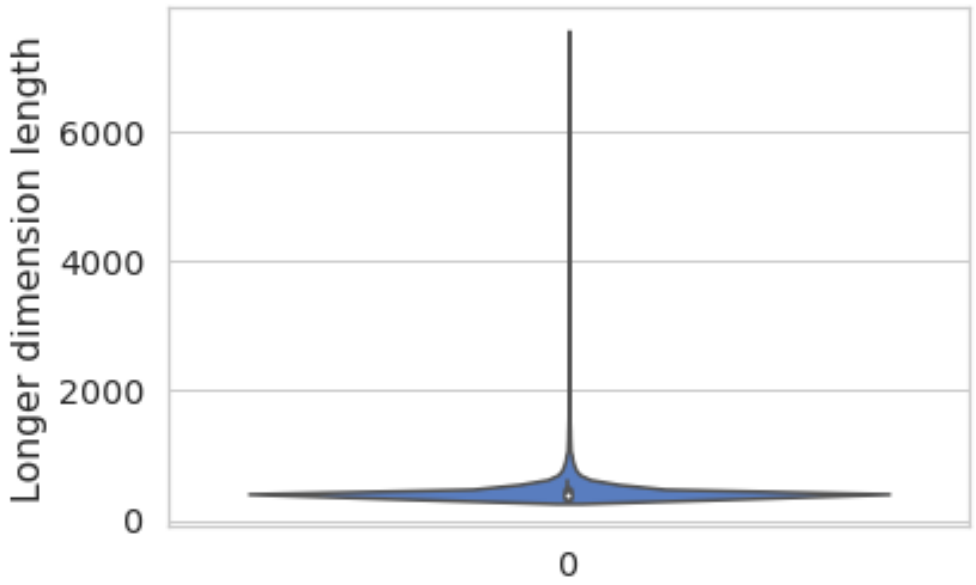

Figure 10: *The length of the longer dimension of the images*

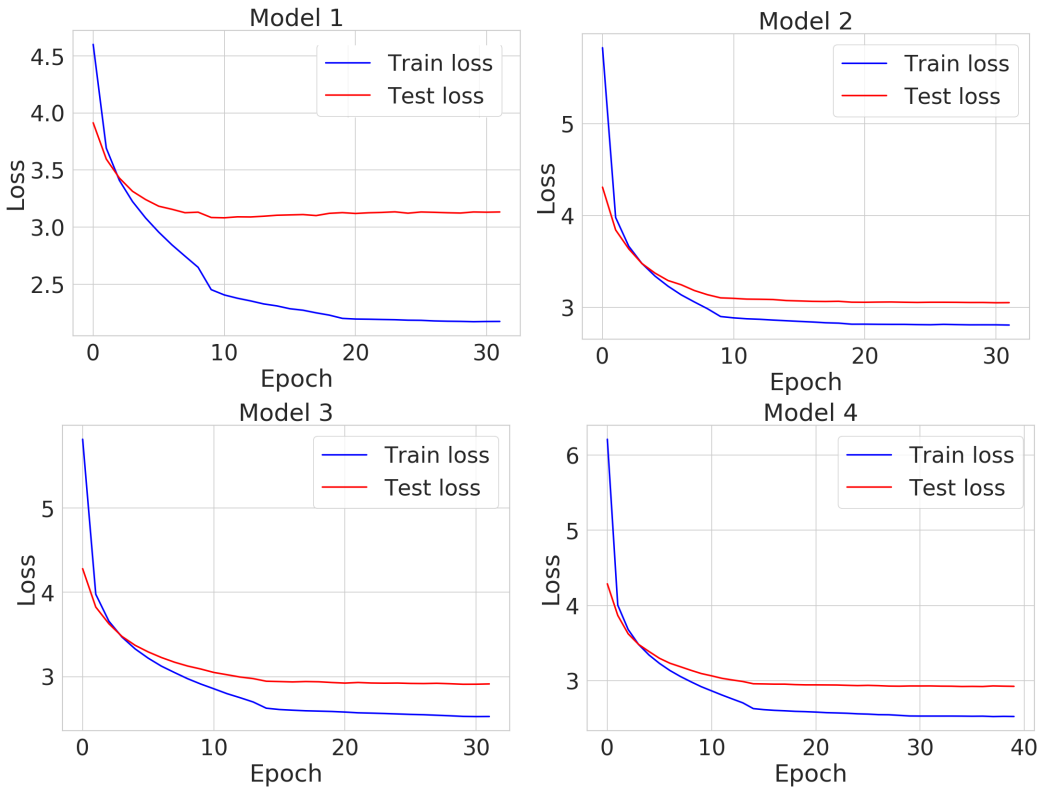

Figure 11: *Train and test errors during training for all 4 models*

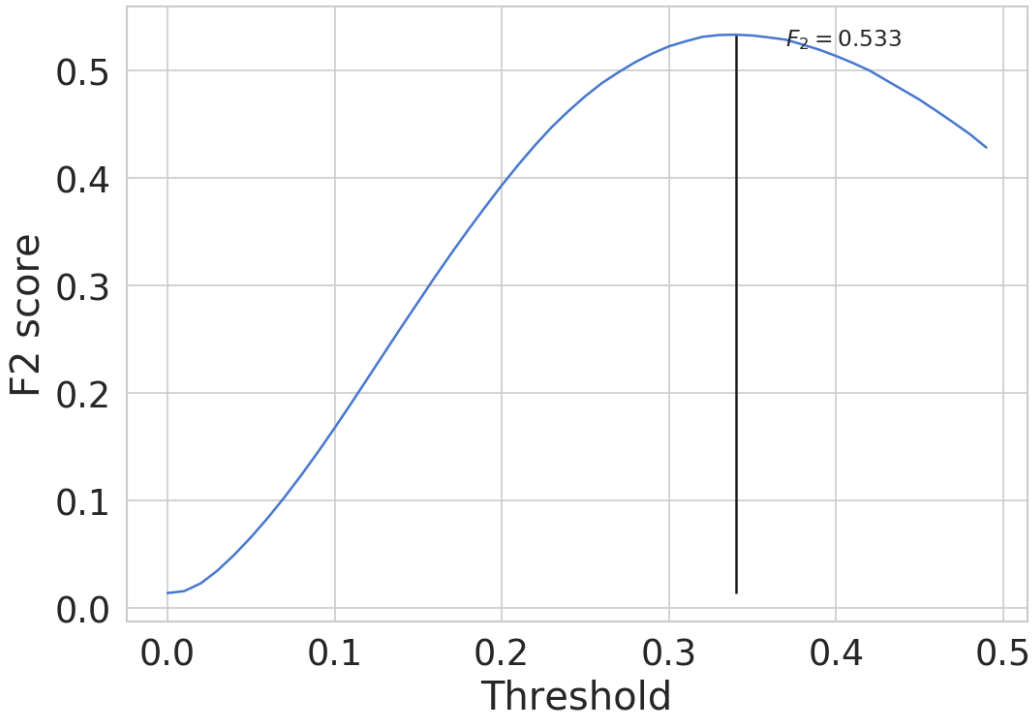

Figure 12: *F2-score vs threshold for model 3*