Machine Learning Engineer Nanodegree

Capstone Project

Classifying Art Pieces Data Set

Gangapalli Venkata Krishna Reddy

December 28th, 2018

Proposal:

Classifying Art Pieces Data Set

I. Definition:

Project Overview:

In this project, I am working on classifying art pieces.

Art has been a part of our life for as long as humanity has existed. For thousands of years people have been creating, looking at, criticizing, and enjoying art. Art is something that captures the eye. Whether the artist is trying to communicate an emotion, an idea or something else, the most important thing is how well the audience receives it. Art is something that inspires people, something that transports us into different realities and moves us into the subconscious places that we did not know existed. Article on Why arts are important for humans???

https://www.enotes.com/homework-help/why-arts-important-humans-today-388402

History: Greeks have extensively used and produced many arts. The Greeks regarded both sciences and crafts as belonging to the realm of art. The Greeks included *music* together with poetry in the sphere of inspiration. (Note has been taken from)

https://sites.google.com/site/encyclopediaofideas/literature-and-the-arts/classification-of-the-arts

When we think on classification, there are several issues to consider. Whether it is really art/not art. There is a really good article on whether something considered as art or not and Correlation with Appraisal and Viewer Interpersonal Differences. This has something more depend on human psychology and creative disciplines. We are not going to dwell into that. Here we are not making any arguments on creativity issues.

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5640778/

https://en.wikipedia.org/wiki/The_arts

here we are clearly making an assumption, everything we encountered here is an ART.

When thinking about this classification of art one thing is important. First, it was concerned not with the products of art but with the act of producing them and in particular the *medium* to produce them.

Based on Medium used for producing,

There are many different types of art like animation, architecture, assemblage, calligraphy, ceramics, computer, Christian or religious, conceptual, artistic design, drawing, folk, graffiti, graphic, illuminated manuscript, illustration, mosaic, painting, performance, photography, sculpture, stained glass, tapestry, and video. Subclassifications include chalk, charcoal, pen and ink, watercolors, acrylics, miniature painting, engraving, lithography, screen printing, wood carving, dance, and acting.

With best advances in Machine Learning like CNN we can clearly, achieve a solution to classify arts based on medium of production.

This solution has very good applications,

A) can used by kids in schooling to Cleary differentiate between arts based on medium. And it is easy to use.
B) Can be used for research purposes on different categories of arts.

We can also expect good performances around 80% accuracy with this project as we are considering only a sub category of different forms of arts.

FUTURE: We can even extend this algorithm to remaining categories.

In this project, I have taken 5 different art forms to classify and their relevant description.

1. Paintings - Painting is the practice of applying paint, pigment, color or other medium to a solid surface **(Wikipedia).**
2. Drawings - Drawing is a form of visual art in which a person uses various drawing instruments to mark paper or another two-dimensional medium **(Wikipedia).**
3. Sculpture - Sculpture is the branch of the visual arts that operates in three dimensions. The earliest example of sculpture dates back to the Upper Paleolithic period (40,000 to 10,000 years ago) **(Wikipedia).**
4. Engravings - Engraving is the practice of incising a design onto a hard, usually flat surface by cutting grooves into it with a burin **(Wikipedia).**
5. Iconography (old Russian art): Iconography, as a branch of art history, studies the identification, description, and the interpretation of the content of images **(Wikipedia).**

Problem Statement:

To understand the difference between types of art forms based on medium. The aim of this project is to predict the type of art form it belongs to by visualizing image. Here I am doing a multi-class classification.

In the project, I am going to use various Machine Learning and Deep Learning Algorithms like NN (Convolutional Neural Networks), CNN (Convolutional Neural Networks) to predict the types of images and compare their performance and finally declare my final model.

Here created model will take an image as input and produce the category of medium is used to draw.

Metrics

I want to use accuracy as evaluation metric for art pieces classification as it is a common metric for categorical classifiers.
Accuracy can be described as
Accuracy = (images correctly classified) / (all images).
As mentioned in capstone proposal feedback, I also added classification report and confusion matrix for all models.
The model which shows prominent feedback on accuracy metric will going to considered as my final model.
I am also intended to plot accuracy and loss of model at different stages of model training for a clear visual deception towards good decision making.

II. Analysis

Data analysis is done in" data analysis" phase of IMAGES Notebook

Data Exploration:

The Classifying art pieces dataset has 9000 richly annotated images;
Total 5 categories of images are present in art pieces dataset, namely

DRAWNGS:

ENGRAVING:

ICONOGRAPHY:

PAINTING:

SCLUPTURE:

The dataset that I am working is downloaded from https://www.kaggle.com/pierrenicolaspiquin/classifying-art-pieces/data. It contains over 9000 images of above mentioned 5 categories.

Citation:

Dataset for classifying different styles of art. Main categories have been taken http://rusmuseumvrm.ru/collections/index.php?lang=en.

The data is open – sourced and can be download for education purpose with no citation.

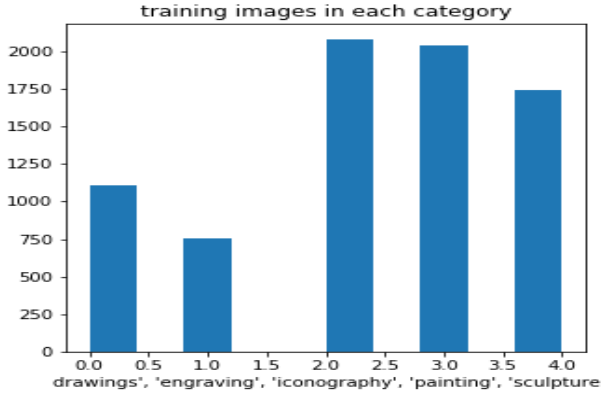
http://rusmuseumvrm.ru/collections/index.php?lang=en.

Data is separated on training and testing sets.

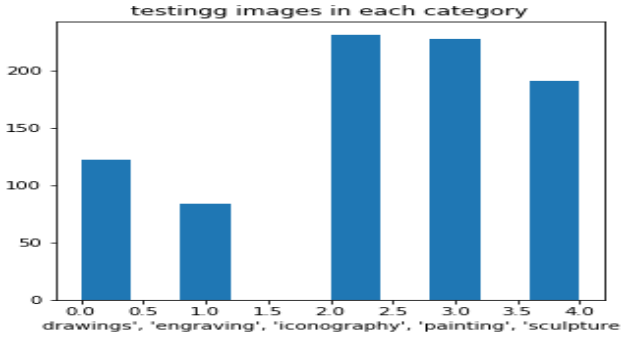
No abnormalities or missing values are found in the dataset.

Exploratory Visualization:

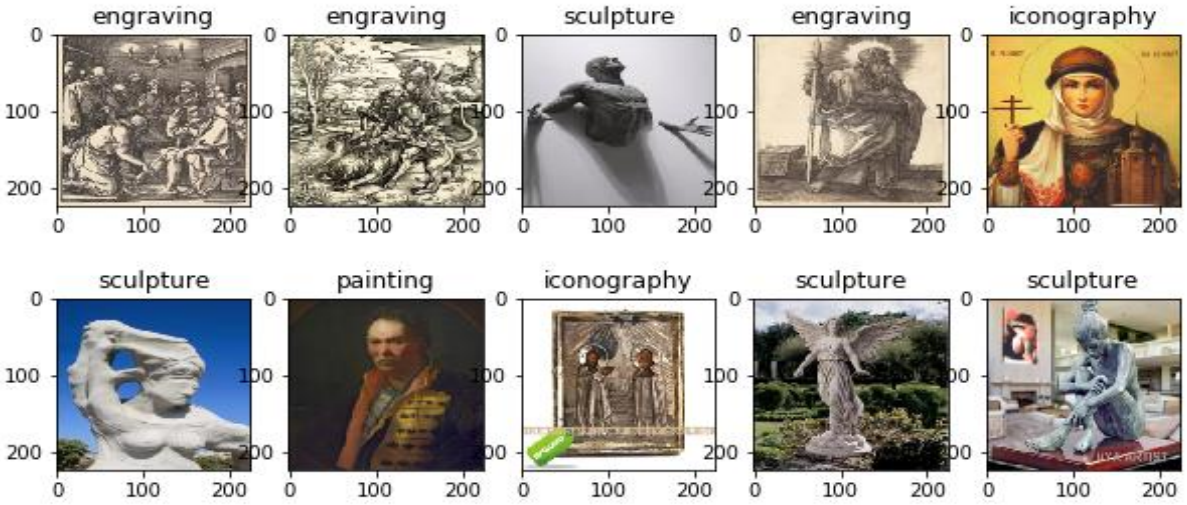
```
There are 5 total art image categories.
There are 8577 total art images.

There are 7721 training art images.
There are 856 testing art images.

Training images in each category.
```



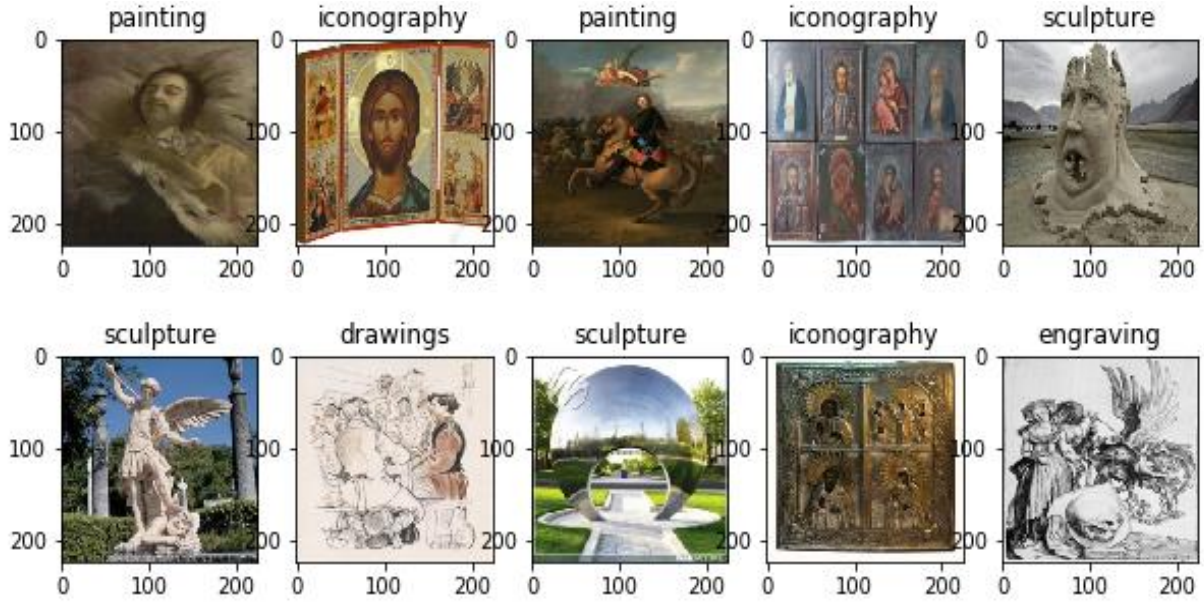
Testing images in each category.



10 sample images in train dataset



10 sample images in train dataset



So far, data is in good way and retrieving successfully.

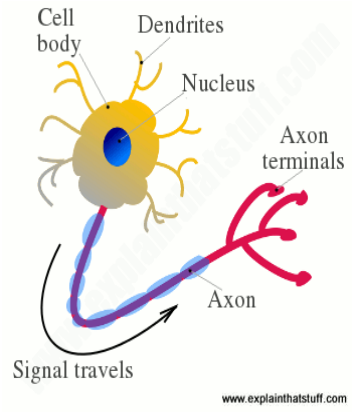
Algorithms and Techniques:

Theory behind Scenes:

Artificial neural networks (ANNs): ANNs are computing systems vaguely inspired by the biological neural networks that constitute animal brains and humans. these systems “learn” to perform tasks by considering examples, generally without being programmed with any task-specific rules.

A typical brain contains something like 100 billion miniscule cells called **neurons** (no-one knows exactly how many there are and estimates go from about 50 billion to as many as 500 billion).

Each neuron is made up of a **cell body** (the central mass of the cell) with a number of connections coming off it: numerous **dendrites** (the cell's inputs—carrying information toward the cell body) and a single **axon** (the cell's output—carrying information away). Neurons are so tiny that you could pack about 100 of their cell bodies into a single millimeter. Inside a computer, the equivalent to a brain cell is a tiny switching device called a transistor. The latest, cutting-edge microprocessors (single-chip computers) contain over 2 billion transistors; even a basic microprocessor has about 50 million transistors, all packed onto an integrated circuit just 25mm square (smaller than a postage stamp)!

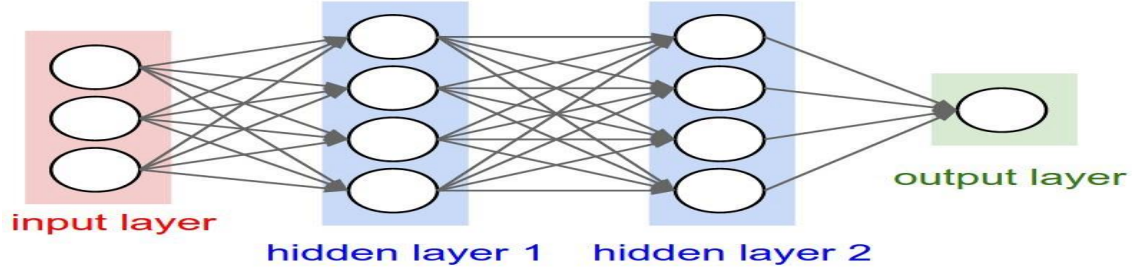


ANN is a set of connected neurons organized in layers:

- **input layer:** brings the initial data into the system for further processing by subsequent layers of artificial neurons. Like dendrites in human neuron
- **hidden layer:** a layer in between input layers and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function. Like nucleus in human neuron.
- **output layer:** the last layer of neurons that produces given outputs for the program. Like axon in human neuron

NOTE: no one how they process information inside.

A typical ANN will look like:



WORKING PROCEDURE:

Step 1- Model initialization:

A random initialization of the model is a common practice. The rationale behind is that from wherever we start, if we are perseverant enough and through an iterative learning process, we can reach the pseudo-ideal model.

Step 2- Forward propagate

The natural step to do after initializing the model at random, is to check its performance.
We start from the input we have, we pass them through the network layer and calculate the actual output of the model straightforwardly.

Step 3- Loss function

At this stage, in one hand, we have the actual output of the randomly initialized neural network. On the other hand, we have the desired output we would like the network to learn. Then we define what we call: **loss function (**for intuition just think loss function like absolute difference or squared difference, but it changes with model to model**)**. Basically, it is a performance metric on how well the NN manages to reach its goal of generating outputs as close as possible to the desired values.

Step 4- Differentiation:

we can use any optimization technique that modifies the internal weights of neural networks in order to minimize the total loss function that we previously defined.

Step 5- Back-propagation:

Then error in loss function is propagated from output layer to input layer by using differentiation we solved in step-4

Step 6- Weight update:

Then corresponding weights in the network are changed in order with learning rate.

*New weight = old weight—Derivative Rate * learning rate*

Step 7- Iterate until convergence:

Just iterate the procedure from step2 to step6 until convergence

NOTE: How many iterations are needed to converge?

- This depends on how strong the learning rate we are applying. High learning rate means faster learning, but with higher chance of instability.
- It depends on the optimization method
- It depends as well on the meta-parameters of the network (how many layers, how complex the non-linear functions are

Now classifier I used is Convolutional Neural Network (one type of Neural Network) , which is the state-of-the-art algorithm for most **image processing tasks**, including classification. It needs a large amount of training data compared to other approaches; fortunately, the art pieces datasets are big enough. The algorithm outputs an assigned probability for each class; by using argmax probability the category for which probability is higher is used as final category of image. The following parameters can be tuned to optimize the classifier:

❖ Training parameters:

➢ Training length (number of epochs)

➢ Batch size (how many images to look at once during a single training step)

❖ Neural network architecture:

➢ Number of layers

➢ Layer types (convolutional, fully-connected, or pooling)

Additionally, along with this I want to experiment with pretrained models like ResNet50 etc.,

Benchmark:

Bench mark model: Any CNN model that gives accuracy around 5% is my benchmark model.

My created model

Conv2D layer with 16 nodes - with in input shape (224, 224 ,3)
Max Pooling layer (aggregation layer)
flatten layer Dense layers with 500 nodes and activation of "relu"
Dropout layer with 0.4 rate Dense layers with 5 output nodes and activation of "SoftMax" layer
At compilation phase loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'] are used.
check pointer is used with 'weights. best. from bench. hdf5', as file path and save_best_only is tuned to True
bench model is trained for 10 epochs 32 as batch size
I declared bench mark model with single convolutional layer and dense layer.

It gives me the training accuracy: 27.22%
It gives me the testing accuracy: 26.9860%

III. Methodology

Data Preprocessing

Data Preprocessing The preprocessing done in the “Prepare data” notebook consists of the following steps:
There are also some preprocessing steps which are done as the images get loaded into memory before
1. The list of images is randomized: to get good redistribution of data when applying split
2. The images are divided into a training set and a validation set (10% of original train set):

To avoid overfitting
3. The images are resized into 3D tensor with shape of width = 224, height = 224, channels = 3 shaped arrays.: so that every image will have same dimensions.
4. after that images are converted from 3D tensor to 4D tensor with shape (1, 224, 224, 3).
5. all images are then normalized by dividing with 255 (all data is now in (0, 1) only)
6. all labels are converted to 5 categories by using keras.utils.to_categorical: in order use for categorical classification.

Implementation:

As in implementation model I have created two models.one is using CNN model and another is using ResNet pretrained model.

Model1: CNN Model

Training phase:

I chose to use a modified version of the Bench mark CNN architecture and add an additional layer while starting with 16 nodes in the first layer as stated with model.
Structure of my CNN model is:

1. To start I have chosen a sequential model as usual
2. A batch normalization layer with predefined input shape

3. Conv2D layer with 16 filters
4. Max Pooling layer (aggregation layer)
5. A batch normalization layer

6. Conv2D layer with 32 nodes
7. Max Pooling layer (aggregation layer)
8. A batch normalization layer

9. Conv2D layer with 64 nodes
10. Max Pooling Layer (aggregation layer)
11. A batch normalization layer
12. Dropout layer

13. Conv2D layer with 128 nodes
14. Max Pooling Layer (aggregation layer)
15. A batch normalization layer
16. Drop out layer

17. Conv2D layer with 256 nodes
18. Max Pooling Layer (aggregation layer)
19. A batch normalization layer
20. Drop out layer

21. Global Average Pooling 2D - Aggregates the previous layers.
22. Sigmoid Output Layer for the 5 categories of images.

I have decided to use CNN model over fully connected model because

- Have lot less parameters, since they share kernel over the patches of whole input image
- Uses feature extraction
- Less parameters less tuning, less tuning less training time and also high accuracy guaranteed.
- A lot can be read through
- https://stats.stackexchange.com/questions/344616/why-use-convolutions-to-image-processing-tasks
- https://stats.stackexchange.com/questions/341863/cnn-vs-fully-connected-network-for-image-recognition

First thing to specify here is as we are using CNN architecture, I want to have at least 3 CNN layers. Because as we know first CNN layer in the model identifies the edges of images. And then second CNN layer in the model identifies shape of images whereas third layer starts differentiating between shapes. (learned in Udacity deep learning). As we working with art images no, particular shapes will be detected like animals in images (Dog Breed Classifier), so I am in thought of add different layers based on accuracy.
Purely on accuracy metric I have add another two CNN layers. As usual output layers with number of categories.

Batch normalization layers:
Batch normalization is a technique for improving the performance and stability of neural networks, and also makes more sophisticated deep learning architectures work in practice
To know more about
https://medium.com/deeper-learning/glossary-of-deep-learning-batch-normalisation-8266dcd2fa82
Max pooling layers:
It is a form of non-linear down-sampling. It works with width and height of the image and performs a down sampling operation on them. as we are increasing the number of dimensions in multiplication of filters in each CNN layer, it is common to use max pooling to reduce the shape of each single dimension. Improves performances in speed.
To know more about how, there is a visual depiction.
https://medium.com/intro-to-artificial-intelligence/simple-image-classification-using-deep-learning-deep-learning-series-2-5e5b89e97926
Drop out layers:
Drop out layers are introduced in last layers because they learn very fast and high when compared with initial layers, so its better to stop from over learning. This will randomly remove nodes in layer. With this we can give chances to weak nodes for learning.
To know more about pls refer Udacity deep learning.
NOTE: No challenges faced when I am training my model, expect some overfitting issues which are simply due to not fitting dropout layers.

Compilation phase:

Trained model has compiled using “rmsprop” as optimizer and 'categorical_crossentropy' as loss function, used metrics is “accuracy”.

check pointer is used with 'weights. best. from scratch. hdf5', as file path and save_best_only is tuned to True
Model is trained by using fit method with parameters 20 as epochs 128 as batch size
While 90% of training data is using for training purpose remaining 10% of data is used for validation purposes.

Refinement:

In this step, there are several refinements are applied to proposed algorithm.
1)after every convolution layer maxpooling layer with pool size =2 is added to reduce the size of each inner layer in model to half.
2) refined my model with batch normalization process. After every maxpooling step one batch normalization layer is added. To speed up the process.
3)also in convolution layers kernel initializer='he normal', activation='elu' are refined to get better performances.
4) to stop model from overfitting, 3 dropout layers are used with percentages 20%, 30 %,
And 40% consecutively after each convolution layer.
5)finally an dense layer with 5 category outputs are refined by using "SoftMax" as activation Function.

MODEL2:ResNet50

Training phase:

As I mentioned in the proposal, though I am satisfactory with my current model, I want to try one of pretrained models on this art dataset.
So, I choose ResNet50 as my pretrained model.
I download ResNet50 model from keras. applications and has given input shape= (224, 224,3)
With parametric tuning weights as "ImageNet". And incude_top as False.
After initializing I add another dense layer with units 1024 and activation function as "elu".
Added a dropout to stop from overfitting as 0.5
Applied batch normalization after dropout layer.
Added a Dense layer with 5 output units and an activation function of "SoftMax" as output layer To ResNet model.

Compilation Phase:

Compiled just as above model1 and also used check pointer to save best weights.
There's nothing to be refined, it is already awesome
Trained ResNet model has compiled using "rmsprop" as optimizer and 'categorical_crossentropy' as loss function, used metrics is "accuracy".
check pointer is used with 'weights. best. from ResNet50. hdf5', as file path and save_best_only is tuned to True
Model is trained by using fit method with parameters 10 as epochs 32 as batch size
While 90% of training data is using for training purpose remaining 10% of data is used for validation purposes.

IV. Results

Model1:
Model Evaluation and Validation

	precision	recall	f1-score	support
drawings	0.67	0.54	0.60	122
engraving	0.69	0.64	0.67	84
iconography	0.90	0.97	0.93	231
painting	0.92	0.92	0.92	228
sculpture	0.85	0.91	0.88	191
avg / total	0.84	0.85	0.84	856

classification report for CNN model to evaluate results.

As we can see that all metrics (precision, recall, f1-score) for iconography, painting, sculpture are above 90%.for remaining two categories also gets above 60% f1-score. From this we can conclude that CNN is robust to unseen data. And can be able to generalize the data pretty well. We can absolutely trust this model for future improvisation. Not only good but it also exceeded my expected outcome of 80%.

Justification

When compared with my benchmark model, my model gives training accuracy up to 90 to 95% Whereas my testing accuracy is around 85%
The results obtained from my model above satisfactory as both training and testing have accuracy scores are above 80% which is pretty good.
Now I can confidently say that my model1 solution significant to solve the problem.

Model2:
Model Evaluation and Validation

	precision	recall	f1-score	support
drawings	0.71	0.59	0.65	122
engraving	0.89	0.58	0.71	84
iconography	0.95	0.93	0.94	231
painting	0.85	0.96	0.90	228
sculpture	0.82	0.93	0.87	191
avg / total	0.85	0.86	0.85	856

classification report for ResNet model to evaluate results.

As we can see that all metrics (precision, recall, f1-score) for iconography, painting, sculpture are above 90%.for remaining two categories also gets above 65% f1-score. From this we can conclude that ResNet50 model is robust to unseen data. And can be able to generalize the data

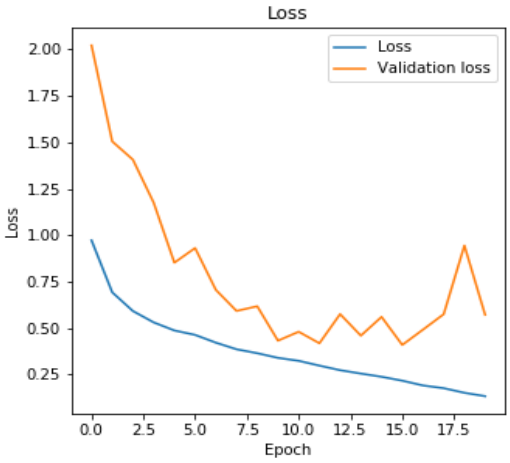
pretty well. If we observe we can see that, ResNet50 have almost equal performance in last three categories (iconography, painting, sculpture) but it shows more promising nature in above two categories (drawings, engraving) than CNN model
We can absolutely trust this model for future improvisation. Not only good but it also exceeded my expected outcome of 80%.

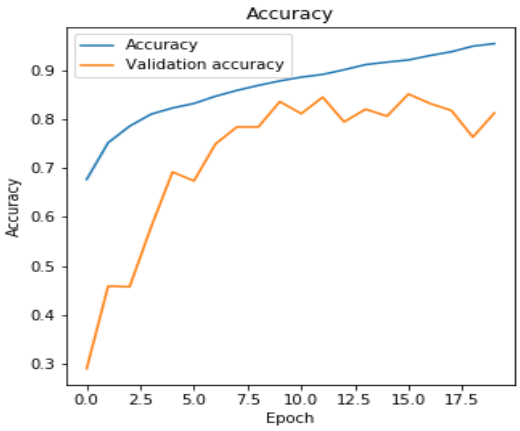
Justification

When compared with my benchmark model, my model gives training accuracy up to 90 to 95% Whereas my testing accuracy is around 85%
The results obtained from my model above satisfactory as both training and testing have accuracy scores are above 80% which is pretty good.
Now I can confidently say that my model2 solution significant to solve the problem.

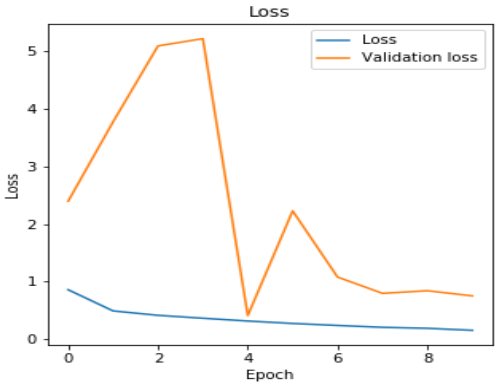
V. Conclusion

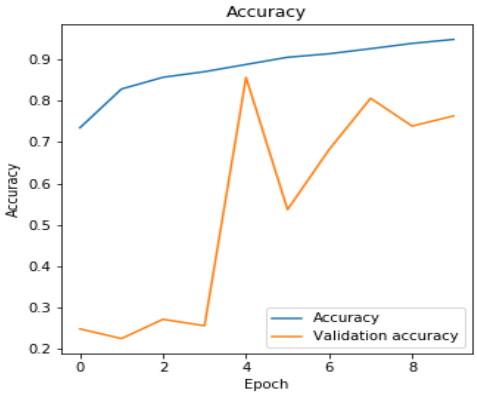
Free-Form Visualization:





These two graphs are my CNN model training history for training data.





These two graphs are my ResNet model training history for training data.

Reflection:

In this capstone project I have taken image classification as my thought of interest inspired from dog breed classifier. In this process I have learned many things.

1) First thing I have learned about data retrieval processes. When I am doing research about retrieval process, I have come across a lot of surprising methods t try. Out of all I decided to use load files method in skleran. datasets
2) Then I used my skills on representation of overall number of images, and also category wise in both training and testing images by using Matplotlib library. I have realized that its most informative in beginning in understanding size of your project.
3) When I stated loading images, I have found many useful methods like image. load_img in keras. preprocessing, cv2.load_img, plt.imread etc., there are lots of them.
4) Then I have learned about using tqdm for reading a list of images.
5) I learn that Image resizing is more important in image classification because we can't expect every image of same size.
6) Then image normalization is used to standardize the all image's (division by 255). I came to know about how numbers in image array changes with brightness. How dark areas will replicate zeros, bright areas replicate 225 etc.,
7) Then I learned that How we divide the data into training and validation using traint_test_split method. Even better split is available as Kfold.
8) Here comes the heart of project, creating a CNN model, fitting it to training data and testing on test data evaluating validation curves, learn from confusion matrix doing modifications on models and there's more going on.
9) Last but not least without visualizing results we can't trust the robustness of a model.

Now coming to overall recap, one best thing about this capstone project it is too scary. But when take image classification like this then it is turned to conjuring.

Improvement

1)One thing, that can be improved from my models is we can use Kfold method for splitting data while fitting model, when we have enough time.
2)we can include many art categories to it
3). Rather than differentiating between different art categories, we can also train model to differentiate between what is model and what is not.

FINAL NOTE:

But, when I trying to improve my two models, I have done two things.
1). Instead of train_test_split method, I have used Kfold split method, soon I came to observe that, both models are taking infinite amount of time to fit training data, so to speak I kind of drop the idea of using it.
2)second and most important thing is image augmentation, when I used augmentation process on my models, both shows training accuracy same as above, but when working with testing, both seems to show less than 30% accuracy, then I came to know that since my data contains only five categories, they both start over fitting. So, I stopped idea of using augmentation. But to speak I learn about augmentation.

Though Both models are giving same performances on training and testing data,

I kind of preferred CNN model, because, its training time is far less than ResNet model,

Resnet model training time exponential to CNN training time. As we see in the free from visualization Resnet starts overfitting after 4 epochs.

So, I consider CNN model as my final model.

Finally, model is subjected to outside sample 3 images from internet.