<u>Machine Learning Engineer Nanodegree</u>

<u>Capstone Project</u>

Image classification of fashion products

iMaterialist challenge (fashion) at FGVC5

Name: Rong-Ching Chang

Date: September 2018

I. Definition

Project Overview

As shopping online became mainstream, precise automatic product recognition is a goal for every e-commerce business. It is one of the critical factors if the consumers are going to purchase, how much market share the business can take and how big the business can grow. E - commerce websites such as Amazon, Flipkart, Snapdeal etc. have millions of products in their inventories. As on January 2018, Amazon was selling over 562 million products. As on August 2018, Flipkart was selling over 30 million products.

Manual distinguishing of similar products, with their diverse classifications and sub categories, could be cumbersome as well as costly. Most of the products sold in the e-commerce space are available in multiple forms, variants, colors, versions and different sets of features.

Employing deep learning techniques to classify products by image accurately, could help businesses developing a more efficient business model. However, automatic product recognition could be challenging:

- Pictures for the same product are taken from different angles, shades, background, intensity of light and cameras.
- Each product may be available in many variants. Their differences could be very subtle such as fuschia pink and hot pink.
- Different products may look similar in shape and color. For example, a bigger size shirt and skirt.

Conference of Computer Vision and Pattern Recognition (CVPR), tackling issues like this, has partnered with Kaggle and held a competition *iMaterialist Challenge (Fashion) at FGVC5* challenging the data scientists, pushing the development of automatic multi label image classification.

CVPR provided the contestants with a dataset consisting of image URLs of the images from 228 numerical labels depicting various fashion product categories. The JSON files provide information about image ID and image URL for fetching the images for training, validation and testing purposes. The training set contains 1,014,544 images, 10,586 images for validation set and 42,596 images for test set.

Problem Statement

Our goal is to classify images of fashion products correctly, assigning the images to one or more numerical labels which represent different fashion product categories.

We have a training dataset of over one million samples. The images would be treated as 3D matrices corresponding to the list of attribute labels. We will train a deep learning model such that when provided with a new image of fashion product, it could tell us the multiple attributes of the product(s) in the image.

To solve this problem, we are going to use a Convolutional Neural Network (CNN) and fastai library for transfer learning. CNNs are deep neural networks which have one or more convolutional layers followed by one or more fully connected layers. CNNs have been delivering impressive results in many deep learning related problems and are greatly used in learning tasks relevant to visual media or image processing. Transfer learning will be used during the training process to classify a given image into the possible classes, product labels in our case. Transfer learning refers to the process of utilizing the weights of a pre-trained neural network, which has already learned to classify, predict, distinguish amongst nuances, to save computational time and deliver better results.

Metrics

We measure the performance of our model using F1 score. To understand how to interpret F1 score, we need to understand what true positive, true negative, false negative and false positive stand for:

	Predicted Class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Negative	True Negative

True Positive (TP) are the correctly predicted positive values, the actual positive class and the predicted class is also positive. In our case, with the assumption that if an image has label [91] it is positive, if the actual class value of our image is [91] and the predicted class tells us [91].

True Positive (FP) are the correctly predicted negative values, the actual negative class and the predicted class is also negative. For example, if the actual class value of our image is not [91] and the predicted class also tells us the predicted value of the image is not [91].

False Negative (TN) are the values which the actual class is yes and the predicted class is no. Such as, the actual class of an image is [91] and the predicted class returns it's not [91].

False Negative (FN) are the values which the actual class is no and the predicted class is yes. Such as, the actual class of an image is not [91] and the predicted class returns it's [91].

After understanding four parameters, we will now take a look at precision and recall:

Precision is the ratio of correctly predicted positive outcomes with respect to the total number of predicted positive outcomes.

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

For example, we have 50 images having the actual class label and predicted class as [91], and 50 images are predicted as label [91] but their actual class is not [91]. Then our precision will be the 50 actual class labels as [91] divided 100 images in total having predicted class as [91], in other word, our precision will be,

$$\text{Precision} = \frac{TP\ (50\ imgs\ act\ class/prd=[91])}{TP(50\ imgs\ act/prd\ class=[91])+FP(50\ imgs\ prd\ class=[91])} = \frac{50}{100} = 0.5$$

Recall is the ratio of the number of correctly predicted positive outcomes with respect to the total number of actual positive class.

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

For example, we have 50 images having the actual class label and predicted class as [91], and 100 images having the predict class as not [91] but their actual class is [91], our precision will be,

$$\text{Precision} = \frac{TP\ (50\ imgs\ act\ class/prd=[91])}{TP(50\ imgs\ act/prd\ class=[91])+FN(50\ imgs\ act\ class=[91])} = \frac{50}{150} = 0.33$$

Having known the precision and recall, the F1 score is calculated as,

$$\text{F1} = \frac{2*Precision*Recall}{Recall+Precision}$$

F1 score is the weighted average of precision and recall. F1 score is useful in situations when false negatives and false positives get penalized differently or uneven class distribution.

To generalize this to multi-class, we use micro average in our case. In “micro averaging,” we calculate the performance, e.g., precision, from the individual true positives and false positives of the k-class model:

$$Precision_{micro} = \frac{TP_1+\cdots+TP_k}{TP_1+\cdots+TP_k+FP_1+\cdots+FP_k}$$

II. Analysis

Data Exploration

The dataset is available as three text files in JSON format for training, validation and testing respectively in the format as shown below,

```
{
        “images”: [
                {
                        “image_id”: <integer value>,
                        “url”: <image URL>
                }, …
        ],
        “annotations”: [
                {
                        “image_id”: <integer value>,
                        “label_id”: [integer values corresponding to labels]
                }, …
        ]
}
```

Data fetching (downloading all images) must be done separately. Each of these images is a 3-dimensional matrix and there can be multiple labels associated with an image, labels are available as integer values, corresponding to the attributes in the image. It should be noted that CVPR chose not to provide labels in the test dataset to avoid contestants training on test set.

There are over a million images provided in the training dataset and there are 228 distinct labels which are used to label these images.

Images in the training set: 1014544

Labels in the training set: 228

Images in the testing set: 39706

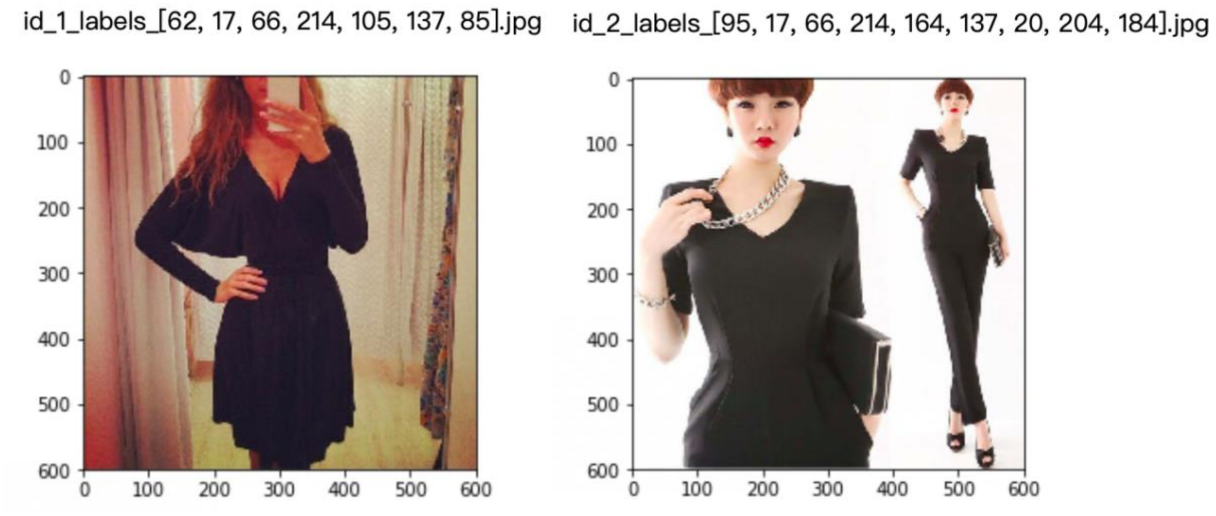
Labels in the testing set: 0

Images in the validation set: 9897

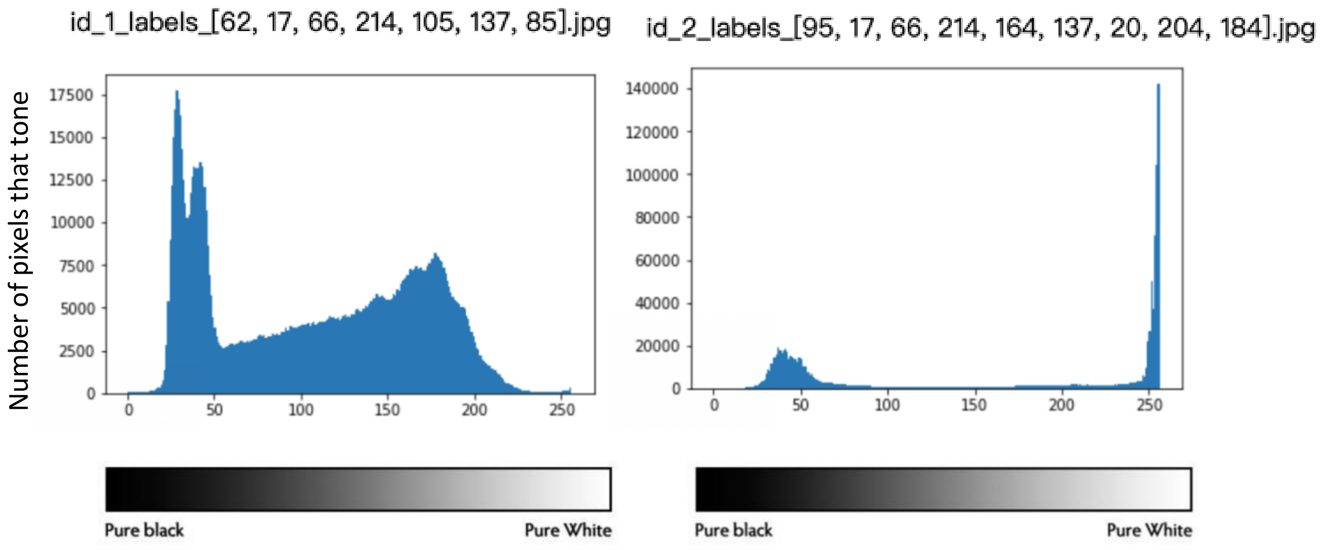
Labels in the validation set: 225

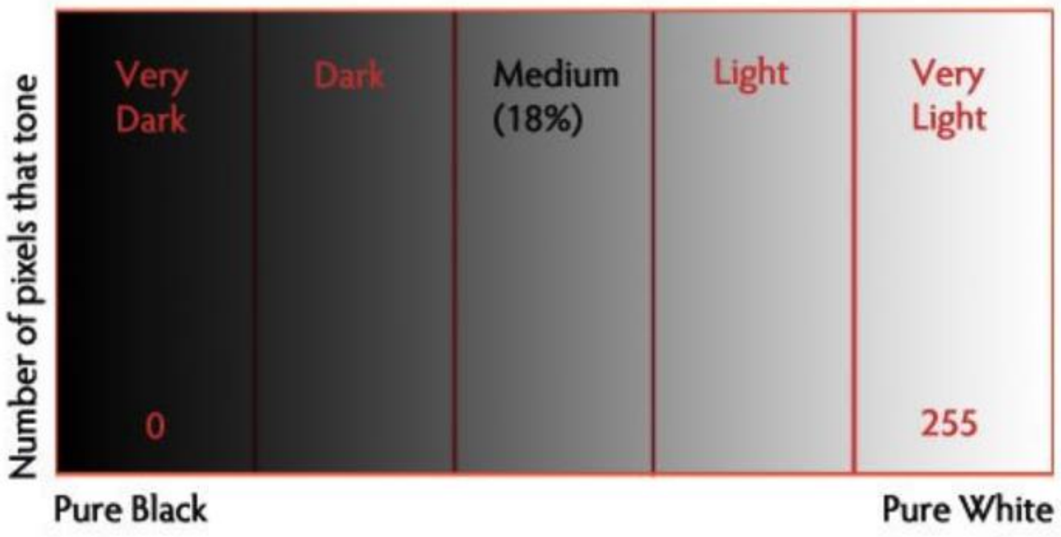
Exploratory Visualization

We will take a quick look of some images from the dataset.

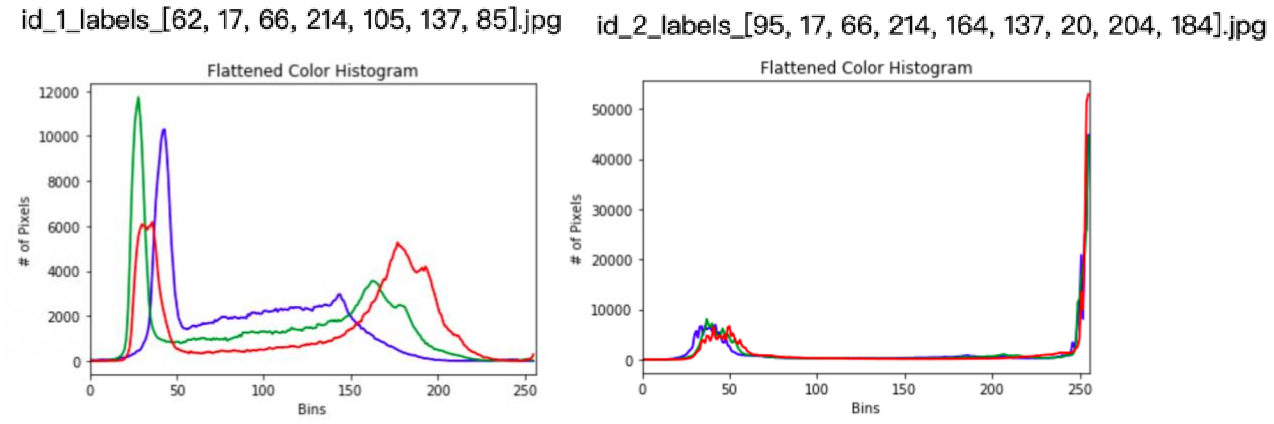

id_1_labels_[62, 17, 66, 214, 105, 137, 85].jpg
id_2_labels_[95, 17, 66, 214, 164, 137, 20, 204, 184].jpg
0
100
200
300
400
500
600

A histogram is a graphical representation of the pixels exposed in an image. In grayscale histogram, x(0,0) represent pure black, x(250,0) is pure white. The middle section between x 0 to 255 is the mid-tones. Y represents the number of pixels in the particular tone. From the histogram, we can't know what objects are there, but we can grasp the characteristics of the pictures, such as brightness, contrast, color temperature, pixel intensity distribution, etc. From the grayscale histogram of image id_1_label.jpg we can assume that there are dark color objects in the image, and the environment of the image is soft light, no bright or strong light in the image. From the grayscale histogram of image id_2_label.jpg we can predict that the image has more contrast color and light than the first image, we have some amount of dark color and extreme amount of light.

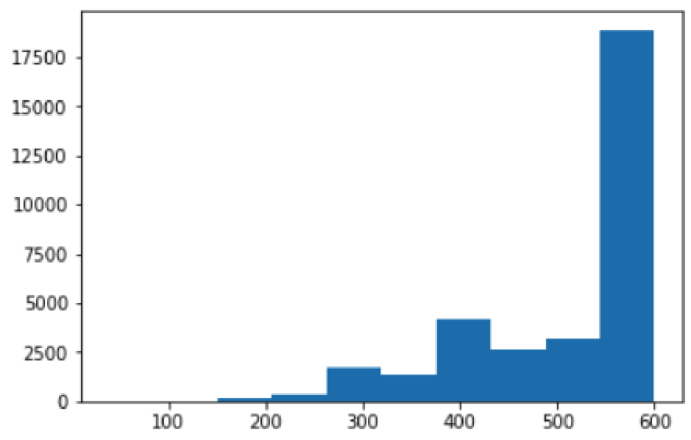




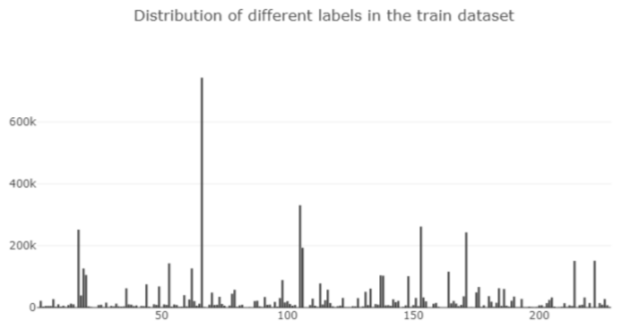
We can also look at the color histogram which represents the color distribution of an image by plotting the frequencies of each pixel values in the r, g, b color channels. We have the same conclusion. In the color histogram of id_1_label.jpg, there are dark color objects, seeing the pick of red color on x(180,0) which means we have some especially light color of red. In the color histogram of id_2_label.jpg, most of the r, g, b color overlaps on the dark and right, showing the color is very simple but also contrast.

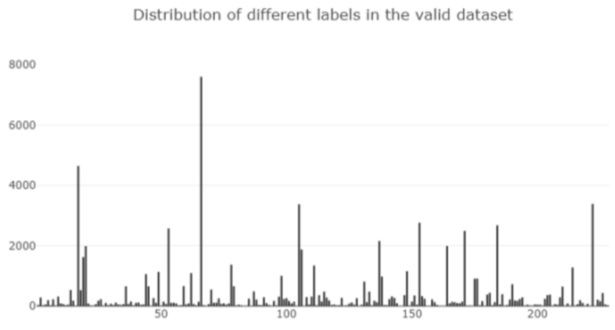


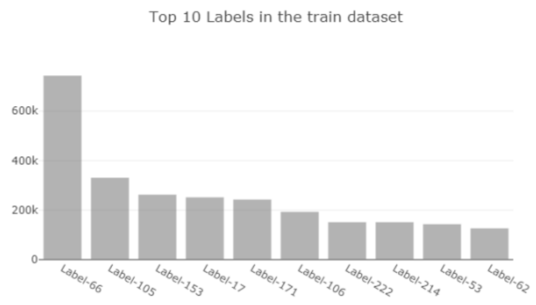
We will look at the first 40,000 images in the training data to get an idea of the image size, we can see that we have a range of size of the image in our dataset. And most of them are 600x600.

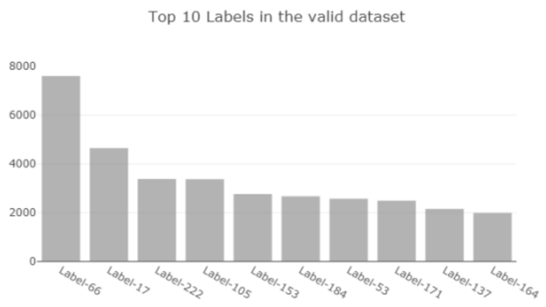


Now we will look at the distribution of different labels in the training and validation dataset. We have uneven number of labels. The most used label is 66 in both training and validation dataset. Also, the top 10 labels in training and validation set are similar.

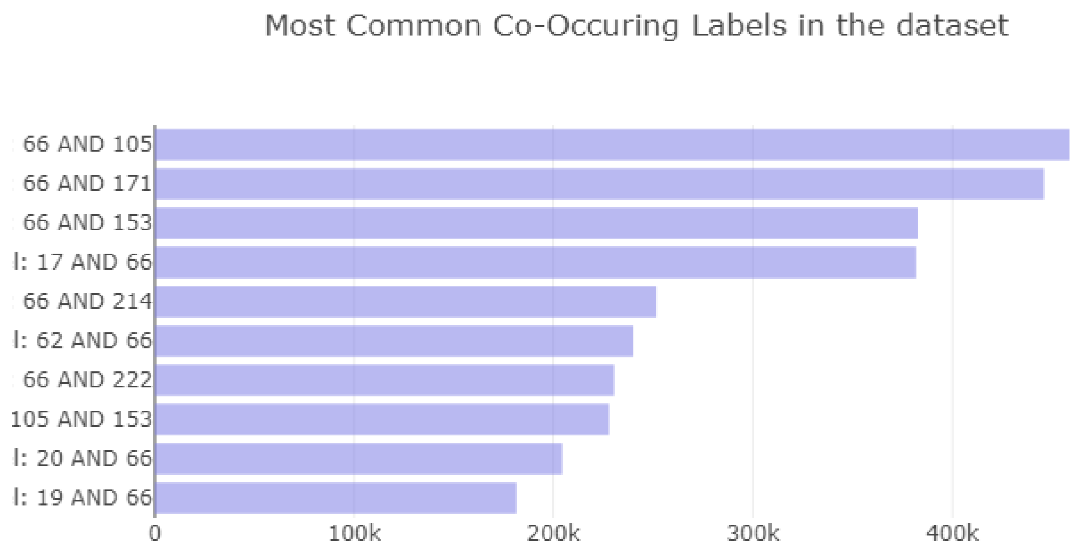








When we look at the most common co-occurring labels in the dataset, most of the co-occurring labels also showed up in the top 10 labels in the training and validation dataset, such as label 66 and 105, 66 and 171, 66 and 153, 66 and 17, 66 and 214, 66 and 222, 105 and 153.

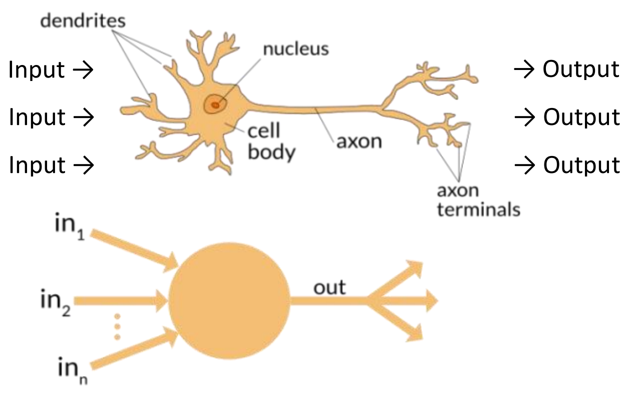


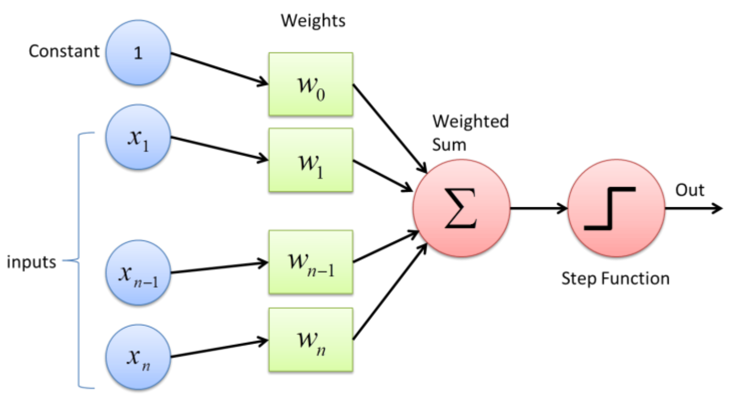
Algorithms and Techniques

Convolutional neural networks are used aided by transfer learning. Fastai provides the state of the art tools for preprocessing, learning, fine tuning and visualizing the results. I have used Fastai's *Resnet 34* deep neural network and transfer learning is applied by using the pre-trained weights available in Fastai's *ConvLearner* module.

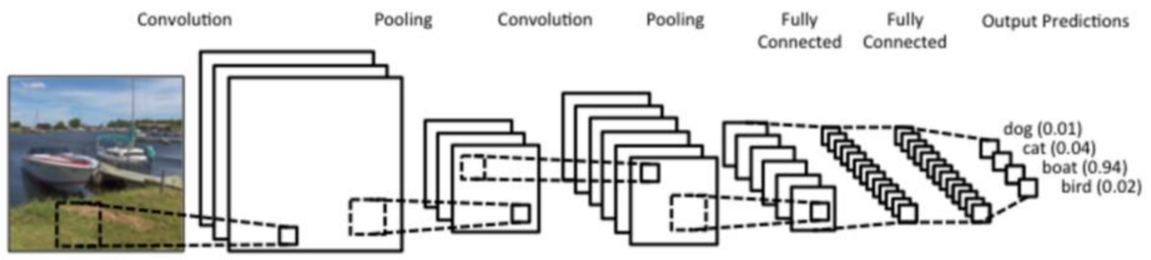
Convolutional Neural Networks (CNNs)

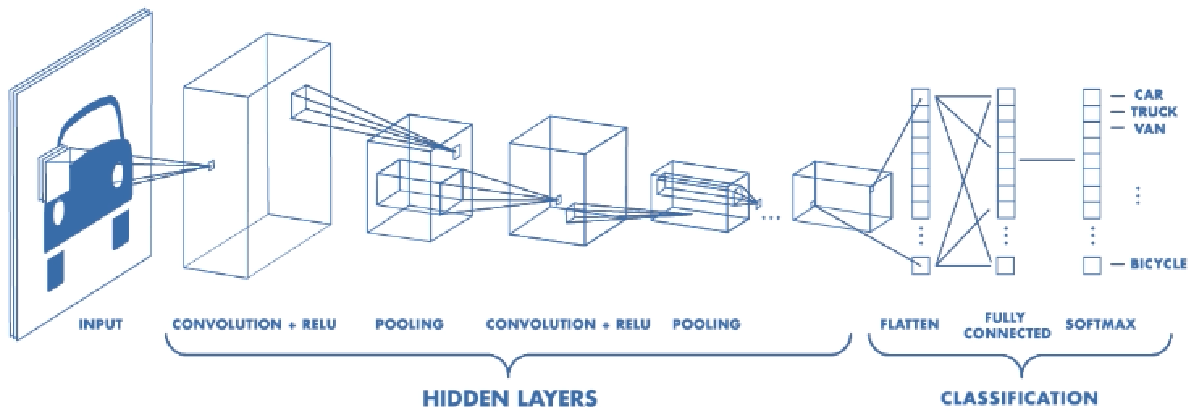
CNNs are one of the models that were inspired by how brain cells work while recognizing objects. Neurons are interconnected nerve cells in the brain that are involved in the processing and transmitting chemical and signals, which is illustrated in the following. Multiple input signals arrive at the dendrites integrated into the cell body, if the accumulated signal exceeds certain threshold, an output signal will be generated and passed on by the axon.



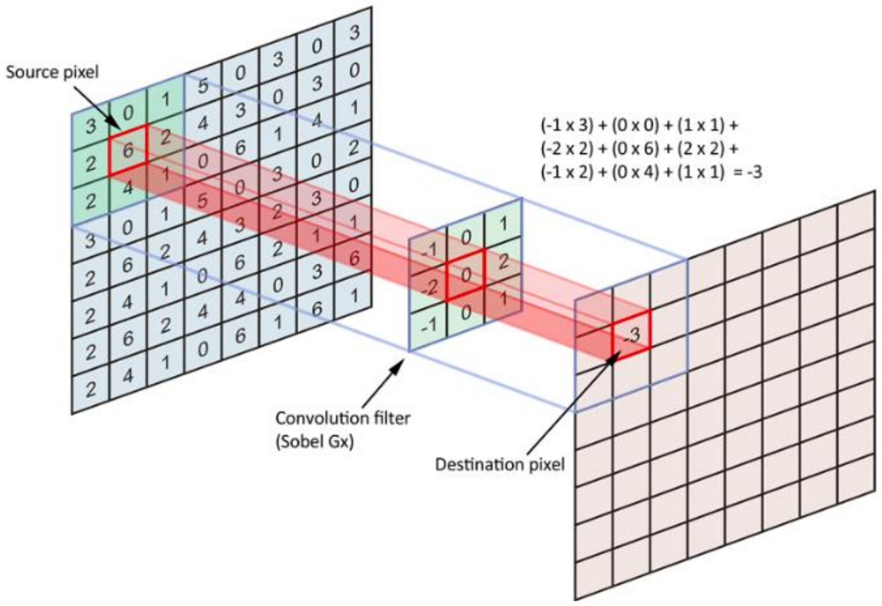


There are some basic architectures that form the CNNs, it contains Convolution, Pooling, and Fully Connected layers.

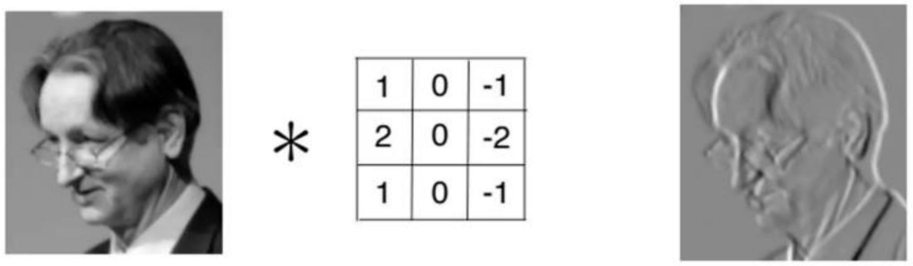




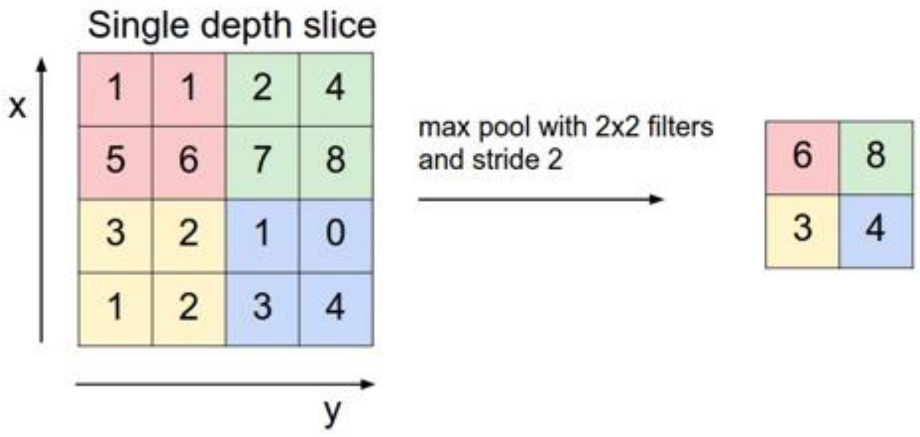
Convolution Layer is the process of creating the dot products between feature detector (also called filter, weight) and input, image is in the form of matrixes. We execute the convolution layer by sliding the filter over the input, every matrix were multiplied and summed and output the result of the feature map.


Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

We use the feature detector to extract the outline of an object.



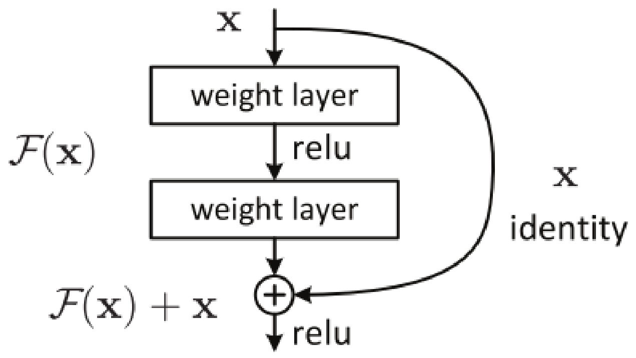
Pooling Layer is added in between convolution layers to reduce dimensionality, number of parameters and computation in the network. It helps shortening the training time, controls overfitting and reduce noise. The most used type of pooling layer is *max pooling*, it takes the maximum value in each slide matrix.



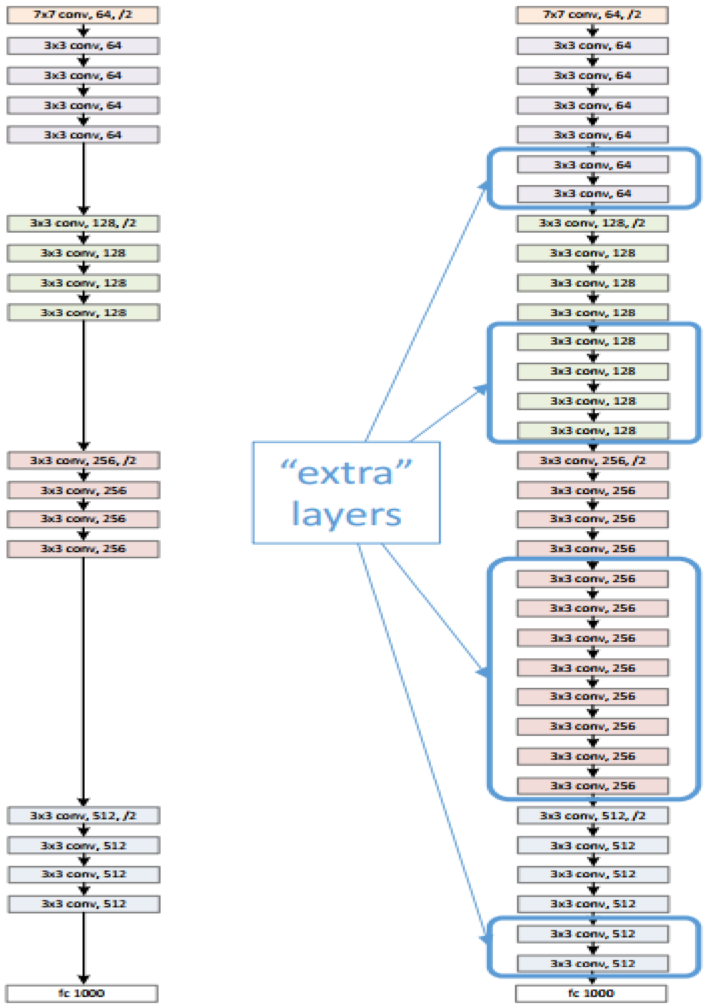
Fully Connected Layer connects all the activations in the previous layer after it's flattened, because it only takes 1 dimensional data. The principle of fully connected layer is the same as a regular neural network.

Resnet 34

The idea behind a Residual Neural Network (ResNet) is that with deep neural networks, the training and test errors spike when the number of layers is increased. However, if the network is made deeper by adding identity layers, the layers which have learned the residuals (errors in prediction) and can be optionally used depending on the input, the accuracy is observed to get better. Shown below is a typical Resnet 34 architecture. We choose resnet 34 because it works well for multi label classification. It needs less training time than other heavy CNNs models.



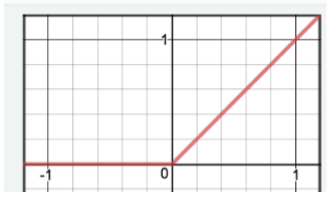
x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu



Transfer Learning refers to the process of using the weights from pretreated network which has been trained on a large dataset to another dataset. Transfer learning is a popular practice in image classification since it requires a lot computing power and can be very expensive. Transfer learning enable us to leverage the results from a well-trained network. I have used the weights provided by Fastai for Resnet 34 and its metrics to fit the training data.

Relu stands for Rectified Linear Units, it is an activation function which aims to apply a non-linear operation to the output of layers. The formula is max(0,z) though it is simple, it provides the benefits of avoiding vanishing gradient problem and it's less computationally expensive than other activation function like sigmoid.

$$R(z) = \begin{Bmatrix} z & z > 0 \\ 0 & z <= 0 \end{Bmatrix}$$



Data Augmentation is a regularization technique which produces multiple images with the same attributes by using different image processing operations such as cropping, applying masks and filters, rotation, rescaling etc. By doing so, we are preserving the labels by altering the pixel intensities. CNNs require a gigantic amount of data to be appropriately trained. Data augmentation helps mitigating overfitting.

Benchmark model

We use pure machine learning as a benchmark model. It contains the basic convolution layer, relu and max pooling layer. We got 0.06 from the result which is much lower than our result 0.6 using rexnet34. The result is as expected, since machine learning needs large training data to get a better result after training, that’s why transfer learning is applied and a popular practice.

III. Methodology

Data Preprocessing

There are a few data preprocessing steps that we performed:

1. Fetching the images
2. Create the csv file
3. Data augmentation

Firstly, fetching the images from the URLs in the training JSON file. Once these images are obtained for training, cross validation and test set, we store them in different pandas data frames and create csv file in the following format, it is how fastai library grabs the image and corresponding labels.

	imageId	labelId
0	train_1.jpeg	95 66 137 70 20

1	train_2.jpeg	36 66 44 214 105 133
	…	

Data augmentation is performed using the aug_tfms=transforms_side_on in tfms_from_model. Considering the character of our images, transforms_side_on takes images and flip them 180 degree. We also randomly zoom in the images with the max_zoom=1.05.

Implementation

We have created a *get_data* function to apply rexnet34 using fastai library. The function *get_data* takes in the size of the image, batch size and the cross validation indexes obtained using *get_cv_idxs* (a NumPy array) to apply transformations on the data to augment it.

```
def get_data(sz):
   tfms = tfms_from_model(f_model, sz, aug_tfms=transforms_side_on, max_zoom=1.05)
   return ImageClassifierData.from_csv(PATH, 'train', 'trn_pro.csv', f'{PATH}trn_pro.csv', bs=bs,
                                        tfms=tfms, val_idxs=val_idxs, test_name='test')
```

Rexnet34 architecture without the last fully connected layer was used first to extract the convolutional features from the preprocessed images. With the Resnet 34 model and the augmented data and the derived F1 score, we begin the transfer learning process with the available pre-trained weights. We loaded the data for augmentation from the CSV file created in a previous step. After some training we unfreeze the rest of layers after training and train more with more layers, and we save the weight. Finally, the predicting process starts for each of the images in the test images in the format described for the competition.

Refinement

We start training with low image size and gradually increase it, such policy helps us building up our weight on rexnet34 and avoid overfitting, we have tested with different image size such as 64, 128, 150, 224, 299. We stopped training if the validation loss start to get bigger than training loss which shows a sign of overfitting. We also tested with a range of different learning rate. We implemented the one cycle policy from Leslie Smith, he suggested using a high learning rates the model can reach high accuracy in much less epochs. However, we didn't get a good result from the one cycle policy, we used a larger learning rate and weight decay. We also used cycle_len and cycle_mult parameters, cycle_len reset the learning rate every given cycle_mult. For example:

```
learn.fit(lr, 3, cycle_len=2, cycle_mult=2)
```

cycle_len = 2 means reset the learning rate every epochs, so our total epochs will be 2^0+2^1+2^2 -1 = 6 epoches.

IV. Results

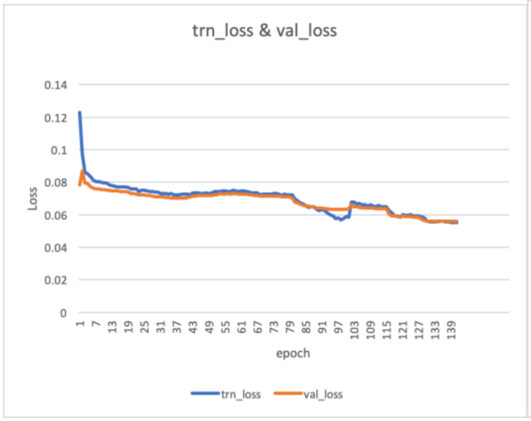
Model Evaluation and Validation

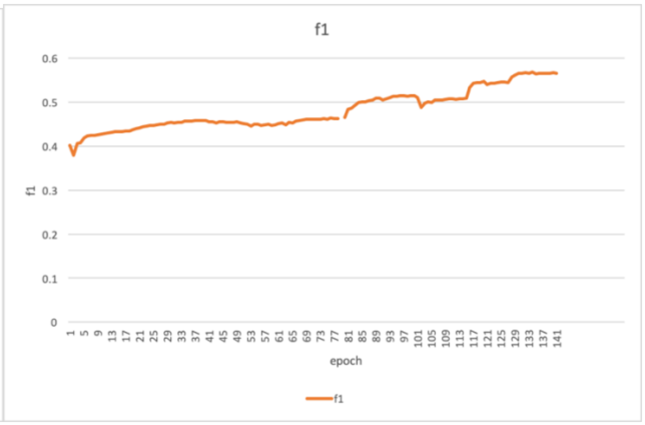
The micro F1 score on the predictions made on the test images is near to 0.6. The range of values the F1 score can have is [0, 1]. The higher the F1 score, the better our model has performed with respect to accuracy. The weights are saved in an epoch if the accuracy outfoxes the value in the previous epoch. Our model used transfer learning with data augmentation and batch normalization to avoid unacceptable amount of variance in our predictions. Fastai gave us the pre-trained weights for Resnet 34 which were used as initial weights. Later, we obtained a better convergence with 30 epochs and the learning rate lowered to a third of the previous value. The preprocessing is handled by Fastai's methods for data augmentation. We used the standard Resnet 34 architecture. Comparing to our benchmark model which is pure machine learning model without using transfer learning, pure machine learning gave us f1 score of 0.06. Transfer learning has proven delivering much better result.

V. Conclusion

Free-Form Visualization

I have recorded part of the f1 and loss of the models per epochs.





As we can see the f1 score is near 60% in the diagram and the loss is near 0.05. we can also see the trend where validation loss and training loss keeps decreasing, when validation loss excessed training loss, we came in and adjust our model by increasing the image size.

Reflection

To reach into this end to end solution, I have thought about various deep neural network architectures and which one could be used in this competition. As the benchmark model, we have tried the pure machine learning model, I also tried heavier model such as resnet___ and non of them deliver the f1 score and efficiency as resnet 34. The images were certainly good, but I was always in a qualm about whether or the given number of images would fare well with my choice of Resnet 34. This wasn’t an ordinary classification problem since there could have been multiple labels for any new image. Currently there is only a few fashion product image classification model on the market but they did not trained on the scale like Resnet34. Thus, in our training, it needs much more images in order to get better result, which means it’s computationally very expensive. We started by using the weight of resnet34 with frozen layer, gradually increase the image size and unfree the rest of the layers on smaller scale of the image number, then we saved the weight and use the whole 1 million images to train.

This was a quite straightforward approach that I took. I got all that was needed with Fastai, used pre-trained weights, extracted CNN features for Resnet 34 for augmenting the data. We choose fastai over keras because fastai have all the tools needed built in to solve large scale multi label image classification problem.

Improvement

It may be evident that I executed my plan with the resources I had. However, I feel that there is scope for improvement in the predictions. Because of limited computational resources, I was unable to try out deeper neural networks which have winning image classification challenges. An alternative to Resnet 34 for the VGG 16 architecture or Google Inception V-3 but in the process of model selection, I realized the problem of the residuals. I would certainly like to try these out in the future.

Citation

Fastai Forum

http://forums.fast.ai

Fastai Github

https://github.com/fastai

1 cycle policy

https://sgugger.github.io/the-1cycle-policy.html

iMaterialist Challenge (Fashion) at FGVC5

https://www.kaggle.com/c/imaterialist-challenge-fashion-2018