Clothing Image Classification

Nirav Sheth

Introduction:

In 1908, the first concept of a electric washing machine was released, and the electric dryer was released shortly after 1938. Since then, we are repeditely completing the next and most tedious task, which is folding clothes. I believe this process can become automated with the help of deep learning image classification models.

However, there are extreme challenges to building the machine. For example, we own a variety of different types of clothes and a variety of different styles of clothes. Due to the variety of clothes, the current folding machines are highly specialized into folding specific types of cloths. The specialized folding machine decreases the market which increases the value of the end product. In the future, there are a few automated folding machine products to be released, but they are more expensive than a typical family can afford.

This project has create a deep learning image classification model to identify common items of clothes, with potential to increase the types of classification as more images are labeled. Potential applications of this model can be automated folding machines, and an automated clothing inventory for businesses and consumers.

Approach:

This project will have 5 main steps:

1. Data collection and importation
2. Preprocessing images
3. Creating the deep learning model using Keras
4. Training and testing the images
5. Implementing model using Webcam

Data:

As of 12/10/2018, I have collected 586 JPEG, PNG and JPG images of dress shirts, dress pants, shorts, and t-shirts. I used a image scraping software to download and filter the pictures that are applicable to this project. Many of the clothing images online have clothes with models, but those images would be additional work for the model that is not needed for this problem. Therefore, I filter through the thousands of pictures scraped from the internet that best fit the model. I scraped from some of my favorite websites like http://us.topman.com,

https://www.jcrew.com/, https://www.everlane.com/, https://www.frankandoak.com in addition to Google and Bing image searches.

Data Importation:

I separated each of the files into 4 separate folders, in which the folders' name will be used as the label. In order to retrieve each of the images' file location, I used the 'GLOB' function that will return the file locations. In order to store the location, I created an array called 'fpaths' and appended each file location into the array. This step will enable to retrieve the label from the file path and the image data.

Data Preprocessing:

This step consisted of resizing, flattening pixels into an array of 'RBG' values, retrieve the label from the file name, and then transpose the array values to fit the model. Each of these methods used the 'OpenCV2' and 'numpy' packages.

Code Snippet used for Data Preprocessing:

```
for fpath in fpaths:
    img = cv2.imread(fpath, flags=1)
    image_resize = cv2.resize(img,(image_size,image_size), interpolation=cv2.INTER_AREA)
    image_list.append(image_resize)
    labels.append(fpath.split('\\')[-2])

data = np.vstack(image_list)
data = data.reshape(-1, 224, 224, 3)
data = data.transpose(0, 3, 1, 2)
```

For the resizing method, I have decreased each of the images to 224 by 224 pixels sizes based on documentation from other popular image classification models (variable 'image_size' which is set to 224). To flattened the images to RGB values in a column format, I first used the OpenCV2 package to read the image into a list, then later used reshape to create an array of 224 by 224 by 3 pixel values for each image. Last step is to transpose the data array into GRB format based on documentation of the VGG16 image learning model.

While the RGB will not be part of the initial scope, I believe it will be important for the versatility of the model (e.g. fabrics, different types of clothing, etc).

Deep Learning Model:

The third step will include creating a deep learning model using Keras package and transfer learning from VGG16 model. The Keras package enables you to create a simple deep learning model over a tensorflow based system. I used the 'ImageNet' model layers and weights to start my classification. I froze everything but the last 4 layers, and then included my few additional layers, so the model can classify the images into my labels. The main steps are VGG16 model, CNN, and pooling. With the model, I implemented the following steps:

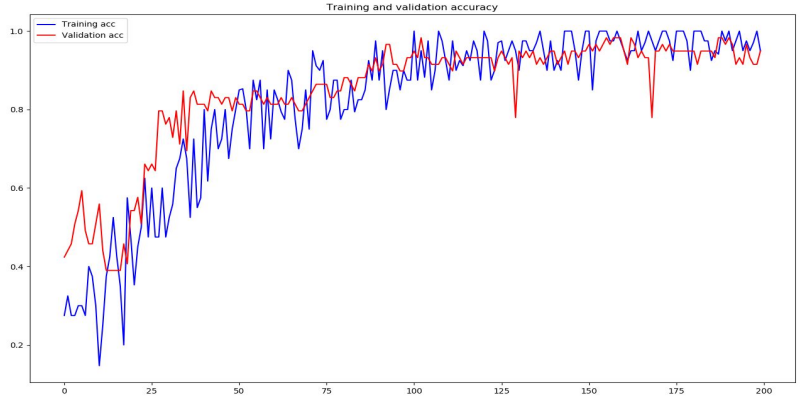
1. Adding VGG16 model with ImageNet weights
2. Additional layers to customize model to my classification needs
3. Decrease overfitting with image augmentation and dropout.
4. Compile with parameters of
    a. Sparce_Categorical_Crossentropy
    b. RMSprop optimization (Root Mean Squared) of a Learning Rate of 1E-4 and decay of 1E-6
    c. Measuring accuracy

Training and Testing Deep Learning Model:

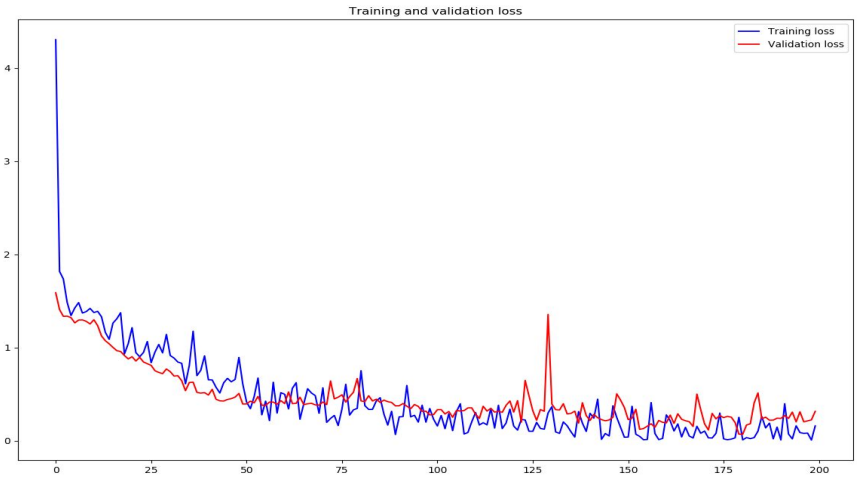
The fourth step will include taking my images and training/test the model. The images will be split into a train, cross validate, test datasets, and then feed into the model. Because I am using my laptop with 8GBs of RAM, I have limited batch size of 8, 200 epochs, 5 steps_per_epochs, and 1 worker.

Results: 97.4% Accuracy and 0.11 Loss

Accuracy over 200 Epochs:



Loss over 200 Epochs:



When deep diving into the data, I found that Dress Shirts had the most incorrect images with Short images with a single incorrect image.

		labels_num
labels	compare	
Dress_Pant_Images	True	150
Dress_Shirt_Images	False	10
	True	136
Short_Images	False	1
	True	80
T_Shirts_images	True	209

Implementation of Deep Learning Model:

Using OpenCV2 and Keras packages, I have loaded my model and implemented OpenCV2 which uses my webcam to take pictures. The pictures will be processed in a similar method as stated above and loaded into the model, which will output a classification label. This output will enable to use the model in a camera based system.

Next Steps:

Firstly, I would like to deep dive into why dress shirts has 10 misclassified images, so I correct this error to make the model even more accurate. Next, I would like to implementing YOLO

image detection model into my model. This will enable to use a video feed to detect objects in real time and efficiently. Further information on YOLO can be found via the link below: https://timebutt.github.io/static/how-to-train-yolov2-to-detect-custom-objects/

Lastly, I would like to create a functioning program that implements this model. For example, I would like to use a webcam system and a conveyor belt to start counting and recognizing cloths.

Proposed Applications:

In addition to the folding machine, other applications for this project:

1. **Clothing Inventory:** I currently live in Houston, Texas where we have faced many hurricanes in the passed. During those times, generous people donate clothes to help the people affected by the storm. However, it takes a lot of volunteers countless hours to count and sort the donations so they can see who they can provide cloths too. I believe this tool could help make this process easier and faster.
2. **Clothing Inventory Part 2:** On the other hand, many of us have tons of clothes in our closet that get lost and forgotten. This tool could help create a clothing inventory, help you quickly look through you virtual closet, and maybe even recommend what to wear.