PROGRESS REPORT 1

Project Name: Neckline Classifier

Period: 30.10.2023 – 27.11.2023

Project Team: Esma Zeynep Sayılı – Esra Cesur

Problem:

Problem 1: In online shopping, large fashion datasets or e-commerce platforms manually tagging and classifiying each clothing can be a time-consuming task. Since consumers will have preferences in the neckline styles.

Problem 2: For marketers and designers, analyzing the fashion trends and consumer preferences is an important detail. Also, manufacterers will need the market demands to produce clothing.

Project Goal and Scope:

The project aims to develop a machine larning model capable of classifying different types of necklines. We are aiming to aid online shopping platforms, designers, manufacterers, and marketers in classifying different types of necklines.

Project Summary:

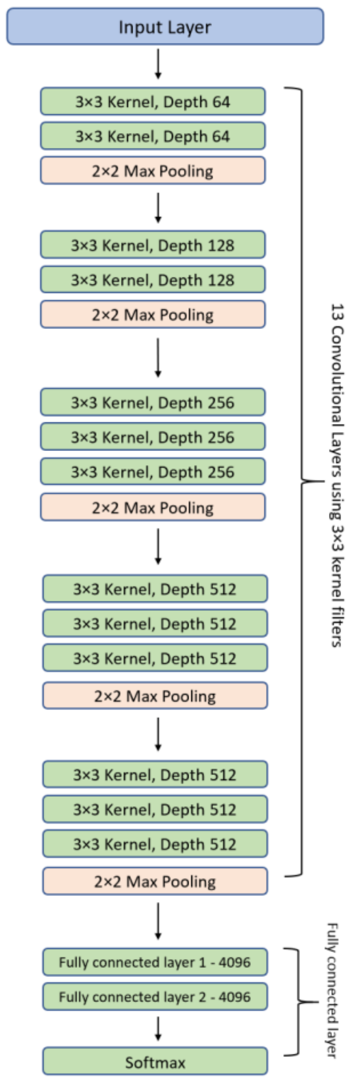
We are using a labeled dataset containing flatlay images. The dataset including images of clothing is labeled in three classes: round neck, scoop neck, and v-neck. We will be using various CNN models to train and test the dataset.

Existing CNN Architectures for Image Classification: A Literature Review

After completing a literature review on the different CNN architectures we can use for our project we gathered the general information on the models we would like to use.

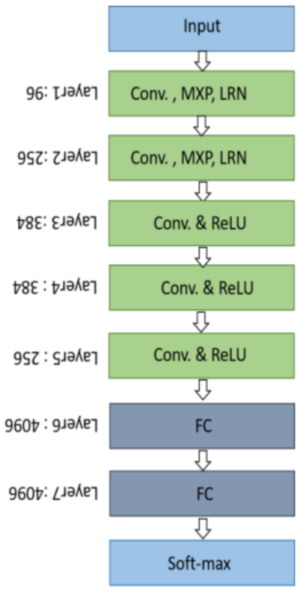
VGG-16 model architecture is a 13 convolutional layers and 2 fully connected layers with 1 SoftMax classifier, in total consisting of 16 layers. The model is introduced with its capability for Large Scale Image Recognition [1].

1. VGG-16 Architecture



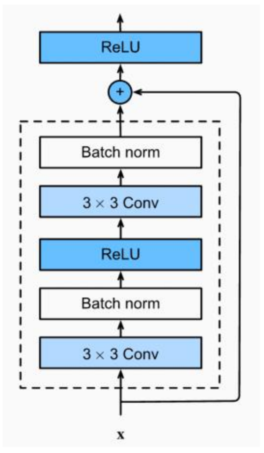
AlexNet model architecture is a 5 convolutional layers and 3 fully connected layers model which performs max pooling with Local Response Normalization (LRN) and a dropout SoftMax layer at the end [2].

2. AlexNet Architecture



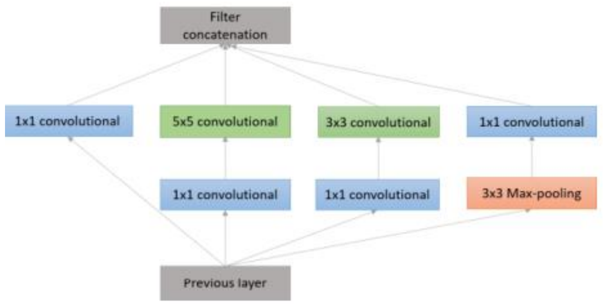
According to the comparison research done by the Harbin Institude Technology VGG16 is confirmed better at removing unrelated background infromation [3].

ResNet model architecture is a Deep Residuel Learning for Image Recognition model. ResNet differs in architecture from VGG and AlexNet models. It is known for using residuel blocks and fits for really deep learning proccesses. The general residuel blocks are implemented as a single convolutional layer [4].

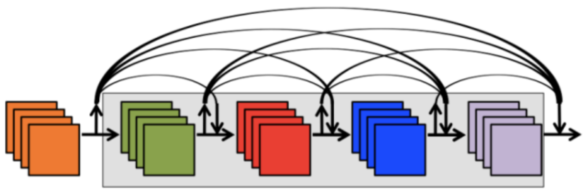


[5]

GoogleNet model architecture is an inception model with inception layers. Inception layers are used to capture diverse features by using different kernel sizes. GoogleNet consists od 22 layers, single layer architecture can be seen in the image below [6].



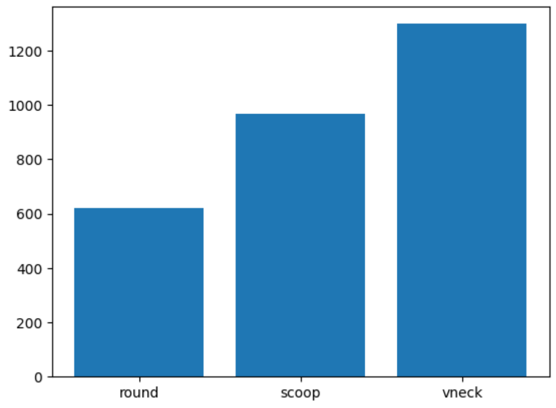
DenseNet is a model architecture that connects the outputs of each previous layers to the current inputs using dense blocks, creating dense connectivity between layers [7].



The Coding Progress:

Dataset Examination:

We analyized our dataset and examined the distribution of classes using Numpy and Matplotlib libraries. Our examination revealed variations in the number of trainable images in our dataset.



The uneven distribution of our classes can affect the performance of our model so we can dealt with this problem using different methods. We can adjust the dataset to have same amount of data from each class by leaving out the excess data or by using weighted loss functions to adjust the weight of each class in order for the minority classes to be as affective as majority classes in the training process.

Preparing the Dataset:

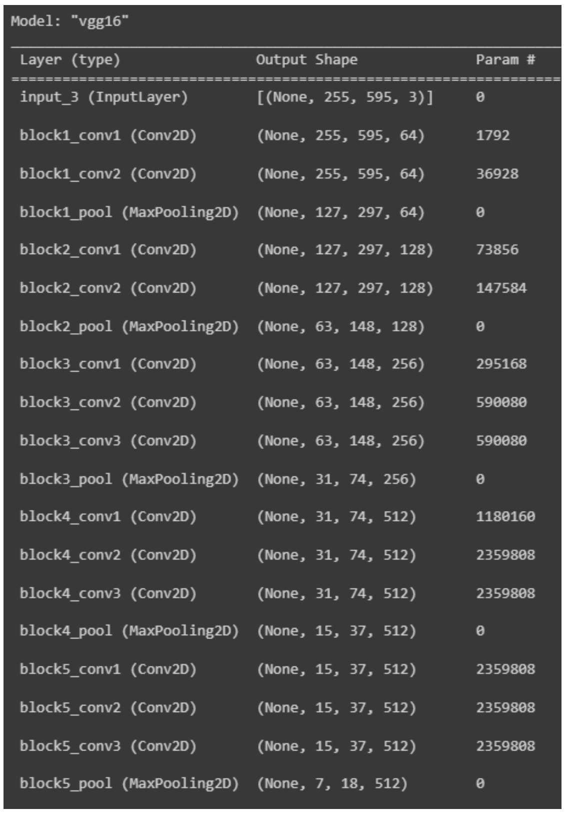
We applied a cropping process on our image dataset in order to focus only on the necklines of the clothing items. During this process we added an additional dimension to our image arrays to include each images class labels. We combined the individual datasets of three different classes into a main dataset. To creates a randomized data we applied a shuffling process using Numpy and Random libraries of Python.

Preprocessing:

Following the preparing of our dataset we divided our array into two array, one housing the clothing images and the other housing the labels. By using the "train_test_split()" function from the "sklearn.model_selection" library, we divided our dataset into training and testing sets. We applied a test split ratio of %33.

Experimenting with VGG-16:

Before starting the whole train and test process we aimed to get ourselves familarized with the models we will be using in our project. Without fitting the dataset to any model, we took our time to understand the parameters, weights and biases of the VGG-16 architecture. We created a model with pre-trained weight and biases on the ImageNet data, with kernel size 3x3. We observed the architecture of our model, and found it identical to the example arcihectures we viewed in our literature review. Our model consists of 14714588 trainable parameters.



```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 255, 595, 3)]     0

 block1_conv1 (Conv2D)       (None, 255, 595, 64)      1792

 block1_conv2 (Conv2D)       (None, 255, 595, 64)      36928

 block1_pool (MaxPooling2D)  (None, 127, 297, 64)      0

 block2_conv1 (Conv2D)       (None, 127, 297, 128)     73856

 block2_conv2 (Conv2D)       (None, 127, 297, 128)     147584

 block2_pool (MaxPooling2D)  (None, 63, 148, 128)      0

 block3_conv1 (Conv2D)       (None, 63, 148, 256)      295168

 block3_conv2 (Conv2D)       (None, 63, 148, 256)      590080

 block3_conv3 (Conv2D)       (None, 63, 148, 256)      590080

 block3_pool (MaxPooling2D)  (None, 31, 74, 256)       0

 block4_conv1 (Conv2D)       (None, 31, 74, 512)       1180160

 block4_conv2 (Conv2D)       (None, 31, 74, 512)       2359808

 block4_conv3 (Conv2D)       (None, 31, 74, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 15, 37, 512)       0

 block5_conv1 (Conv2D)       (None, 15, 37, 512)       2359808

 block5_conv2 (Conv2D)       (None, 15, 37, 512)       2359808

 block5_conv3 (Conv2D)       (None, 15, 37, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 18, 512)        0
```

REFERENCE

[1] Srikant Tammina. (2019). "Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images". International Journal of Scientific and Research Publications (IJSRP). url:https://www.researchgate.net/publication/337105858_Transfer_learning_using_VGG-16_with_Deep_Convolutional_Neural_Network_for_Classifying_Images

[2], [6], [7] Md Zahangir Alom1 , Tarek M. Taha1, Chris Yakopcic1, Stefan

Westberg1, Paheding Sidike2, Mst Shamima Nasrin1, Brian C Van Essen3, Abdul A S. Awwal3, and Vijayan K. Asari. (2018). “The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches”. Computer Vision and Pattern Recognition. Cornell University. url:https://arxiv.org/abs/1803.01164

[3] Wei Yu, Kuiyuan Yang, Yalong Bai, Tianjun Xiao, Hongxun Yao, Yong Rui. (2016, June). “Visualizing and comparing AlexNet and VGG using deconvolutional layers”. In Proceedings of the 33rd International Conference on Machine Learning. url:https://icmlviz.github.io/icmlviz2016/assets/papers/4.pdf

[4] Sasha Targ, Diogo Almeida, Kevin Lyman. (2016). “Resnet In Resnet: Generalizing Residual Architectures”. Machine Learning. url:https://arxiv.org/abs/1603.08029

[5] Yashowardhan Shinde. (2021). “How to code your ResNet from scratch in Tenserflow?”. DataScience.

url:https://www.analyticsvidhya.com/blog/2021/08/how-to-code-your-resnet-from-scratch-in-tensorflow/