EE 58J Data Mining Project 4

Oğuzhan Sevim
June 2, 2020

I. INTRODUCTION

In the final part of Product Image Recognition challenge, we implement some of the data augmentation methods and the convolutional neural network architecture called $VGG-16$ on the Vispera -SKU$101-2019$ dataset. We will use a $VGG-16$ network which was already trained on large dataset of ImageNet challenge. However, instead of using its last 3 FC layers, we use only 2 FC layers in our model. The model will be trained by fine-tuning these last 2 layers.

II. DATA

Vispera-SKU$101-2019$ data is composed of approximately 10.000 arbitrary–sized images in jpeg format. Each image belongs to one of 5 categories where each category contains around 20 classes. In this project, we will separately experiment with 2 categories: confectionery and ice cream. An example image from ice cream category with class name of sku.415 is shown in Figure 1.

Fig. 1. An example image from ice cream category with class name sku.415

III. DATA AUGMENTATION

When the dataset is small compared to dimensions of feature space, the model can't learn about most parts of the feature space. This results in poor generalization. In order to make the model generalize well on the new data, we expand the training dataset by using technique called data augmentation. In this project, *imgaug* library is used for the augmentation, and dataset is expanded by generating 5 new samples out of each training sample. Each of these 5 samples are created by using one of the following methods:

1) Horizontal flipping.
2) Adding Gaussian noise on the image: The noise on each pixel and channel is independently sampled from normal dist. of $\mathcal{N}(0, 0.25 * 255)$.
3) Random cropping: We crop images from each side by 0 to 50px (randomly chosen).
4) Affine transformation: Rotate (between -45 to 45 degrees) and shear (from -32 to 32 degrees) each image.
5) Change the color and brightness: Each image is first converted to HSV and random values (between -50 to 50) are added on H-S channels. Then, change V of HSV.

When these 5 augmentations are implemented on the image shown in Figure 1, the new images are generated as shown in Figure 2.

Fig. 2. New images generated randomly (except the first one) from Image 1 by using methods listed above

IV. VGG-16 MODEL

VGG-16 is a convolutional neural network model which consists of 13 convolutional, 5 max-pooling, and 3 fully connected layers. The architecture of the model is shown in Figure 3. In our implementation, last 3 FC (dense) layers are replaced with 2 dense layers that have 128 and 20 units respectively.

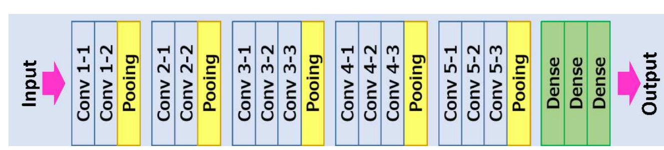

Fig. 3. VGG-16 model architecture

V. FIRST ATTEMPT

In the first attempt, we only trained the last 2 dense layers of our model. For the convolutional layers, weights of a model trained on ImageNet dataset are directly used and kept the same.

In order to prevent overfitting, early stopping is used. In the training process, we can easily make Keras stop training when a monitored metric has stopped improving. For that purpose, I portioned 20% of the training dataset (if augmentation is used, portioning is applied on augmented data) as validation data and monitored its accuracy. If the validation accuracy stops increasing for 3 epochs (patience), the training process simply gets terminated. For the activations of the dense layers, ReLU and Softmax functions are chosen. Finally, Adam optimizer is used in the training.

Results of the first attempt are given in Sections V-A and V-B.

A. Confectionery Category

For confectionery data, I first implemented 2 models that respectively have $64-20$ and $128-20$ nodes in the dense layers. For these layer width choices, accuracies shown in Tables I and II are obtained. From these tables, the $128-20$ configuration seems to be slightly better. Thus, I will only implement $128-20$ model in the ice cream category.

TABLE I
FIRST ATTEMPT ACCURACIES ON CONFECTIONERY CATEGORY WHEN THE LAST 2 LAYERS HAVE 64 AND 20 UNITS.

Configuration	Test Acc.	Tra. Acc.
Baseline10	32.71	98.78
Augmented10	51.59	100
Baseline20	42.37	99.39
Augmented20	62.89	100
Baseline50	59.35	100
Augmented50	72.31	99.92

TABLE II
FIRST ATTEMPT ACCURACIES ON CONFECTIONERY CATEGORY WHEN THE LAST 2 LAYERS HAVE 128 AND 20 UNITS.

Configuration	Test Acc.	Tra. Acc.
Baseline10	36.09	95.12
Augmented10	50.03	99.9
Baseline20	46.25	100
Augmented20	62.89	100
Baseline50	60.21	100
Augmented50	73.67	100

B. Ice Cream Category

When we implement the model that have $128-20$ units in the dense layers, we get the results as shown in Table III.

TABLE III
FIRST ATTEMPT ACCURACIES ON ICE CREAM CATEGORY WHEN THE LAST 2 LAYERS HAVE 128 AND 21 UNITS.

Configuration	Test Acc.	Tra. Acc.
Baseline10	39.23	98.83
Augmented10	57.03	100
Baseline20	50.5	100
Augmented20	66.96	100
Baseline50	64.15	100
Augmented50	76.1	100

VI. SECOND ATTEMPT

Obviously, the results shown in Tables I-II-III are not satisfactory in terms of generalization on the unseen data. I forgot to include the validation accuracies in the Tables, but they were always between $90\% - 100\%$. Since I used early stopping on the validation data, training the models for further epochs would only decrease the validation accuracies, and the model would overfit. That was a sign of the need for unfreezing some of the earlier layers.

In the second attempt, I unfreeze the last convolutional layer and train it with the dense layers. By doing that number of trainable parameters are increased from $68k$ to $2.5M$. To prevent overfitting, the following regularization methods are implemented:

- $L2-$regularization with $\alpha = 0.01$.
- Dropout with $p = 0.2$ after the dense layer with 128 nodes.
- Early stopping by monitoring validation accuracy.

The results of this setup are given Tables IV and V.

TABLE IV
SECOND ATTEMPT ACCURACIES ON CONFECTIONERY CATEGORY.

Configuration	Test Acc.	Tra. Acc.	Val. Acc.
Baseline10	43.57	96.95	50
Augmented10	62.02	100	93.95
Baseline20	58.96	100	56.63
Augmented20	72.58	99.95	93.75
Baseline50	76.38	99.88	77.78
Augmented50	81.51	99.15	90.96

TABLE V
SECOND ATTEMPT ACCURACIES ON ICE CREAM CATEGORY.

Configuration	Test Acc.	Tra. Acc.	Val. Acc.
Baseline10	50.65	100	53.49
Augmented10	59.37	98.44	90.66
Baseline20	54.7	88.6	51.16
Augmented20	76.07	99.7	91.83
Baseline50	73.11	94.39	72.43
Augmented50	83.66	99.75	93.69

VII. FINAL THOUGHTS AND ACKNOWLEDGEMENTS

At the beginning, I implemented Inception_$v3$ model. However, after training the model on a portion of the dataset (augmented or not), the final accuracy of the training history did not match the accuracy of the model when I tested it with the training data. After searching online, I saw many people encountered with the same problem which was caused by the batch normalization blocks of the network. Therefore, I decided to use a batch normalization free model (e.g. VGG-16 or VGG-19).

I experienced well the difficulties of creating deep networks. For me, biggest problem was the demand for high computational power. Even training few dense layers took hours, and experimenting with different hyperparameters was difficult. I would like to experiment more with different validation set sizes, activations, optimizers, layers sizes, etc., but my computational resources did not allow me to do that. However, I also have to admit that I made a mistake about parameter tuning. I could have experimented more by only using a single configuration (e.g. Augmented20) to find the best setting. Then, I could implement the best settings on the rest of the configurations. I believe that I could achieve higher accuracies ($> 90\%$) by doing so.

Lastly, I think I have acquired great practical and theoretical skills in this course. Thank you for this nice course.