Task 5

Yahya Ramzy [02-SBIRfeatureX]

15 April 2023

1 VGG-16 CNN Model

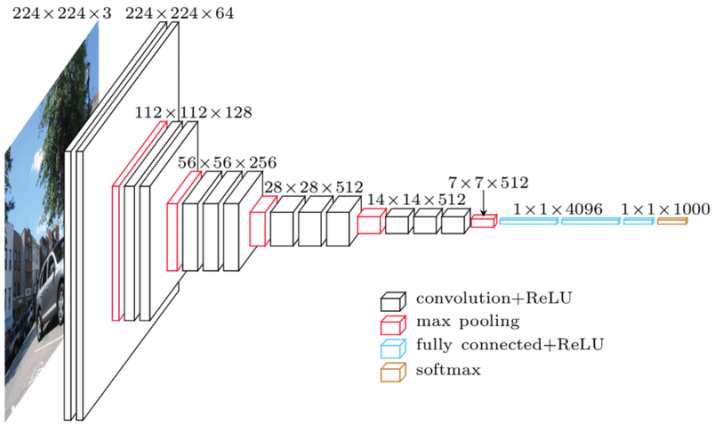

Figure 1: VGG-16 CNN Architecture

1.1 What is a VGG-16 Model?

The VGG16 CNN model is a deep neural network architecture that was introduced by Simonyan and Zisserman in 2014 [1] for image classification tasks. The model consists of 16 layers, including 13 convolutional layers and 3 fully connected layers, and it is characterized by its use of small 3x3 filters in each convolutional layer, which helps to capture finer details in images. The VGG16 model achieved state-of-the-art results on the ImageNet dataset, and has been widely used as a benchmark for image classification tasks in computer vision research.

1.2 Why use a VGG-16 Model?

The VGG16 CNN model has been shown to be effective in extracting high-level features from images, which makes it a promising candidate for sketch-based image retrieval tasks. Sketches often lack the rich visual details present in natural images, which can make traditional image retrieval techniques less effective. However, by utilizing the deep convolutional layers of the VGG16 model, features can be extracted from sketches that capture important information about shape, texture, and overall structure. These features can then be used to compare and retrieve images that match the sketch, making the VGG16 model a valuable tool for sketch-based image retrieval applications.

Moreover, the VGG16 model has been proven to be a transferable feature extractor in many image-related tasks, which enables us to apply this model in various domains with only a small amount of fine-tuning. For instance, in a study [2], the VGG16 model was adapted to the sketch-based image retrieval task in the domain of remote sensing images. The results demonstrated that the VGG16 model achieved significant improvements in retrieval accuracy, compared to other traditional and deep learning-based retrieval methods. These findings show that the VGG16 model is not only effective in natural images but also in domains such as remote sensing, where the extracted features can be utilized for image retrieval tasks.

1.3 Training a VGG-16 Model with our dataset.

1.3.1 Data Preprocessing

Before training the VGG16 model, we preprocessed the dataset by resizing all images to 224x224 pixels and normalizing the pixel values to be in the range [0, 1]. We also applied data augmentation techniques to the training set, including random zooming, shifting, and shearing, to increase the size of the training set and reduce overfitting.

Method	Setting
Resize	224x224
Normalization	[(0,255)→(0,1)]
Zoom Range	0.15
Width Shift Range	0.2
Height Shift Range	0.2
Shear Range	0.15

Figure 2: Methods used in Data Preprocessing and their Settings for VGG16

1.3.2 VGG16 Model Architecture

The VGG16 model consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. The convolutional layers are grouped into five blocks, with each block consisting of two or three convolutional layers followed by a max pooling layer. The fully connected layers serve as a classifier and map the feature vectors extracted by the convolutional layers to the output classes. We used the

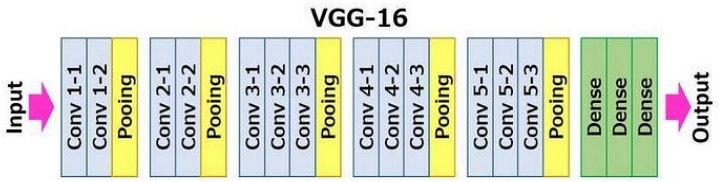

Figure 3: VGG-16 Architecture Flow Chart

Keras framework to implement the VGG16 model with pretrained weights. Specifically, we used the VGG16 model with the ImageNet weights, which were pretrained on a large-scale image recognition dataset with 1,000 object categories.

Layer (type)	Output Shape	Param #
Conv2D	(None, 224, 224, 64)	1,792
Conv2D	(None, 224, 224, 64)	36,928
MaxPooling2D	(None, 112, 112, 64)	0
Conv2D	(None, 112, 112, 128)	73,856
Conv2D	(None, 112, 112, 128)	147,584
MaxPooling2D	(None, 56, 56, 128)	0
Conv2D	(None, 56, 56, 256)	295,168
Conv2D	(None, 56, 56, 256)	590,080
Conv2D	(None, 56, 56, 256)	590,080
MaxPooling2D	(None, 28, 28, 256)	0
Conv2D	(None, 28, 28, 512)	1,180,160
Conv2D	(None, 28, 28, 512)	2,359,808
Conv2D	(None, 28, 28, 512)	2,359,808
MaxPooling2D	(None, 14, 14, 512)	0
Conv2D	(None, 14, 14, 512)	2,359,808
Conv2D	(None, 14, 14, 512)	2,359,808
Conv2D	(None, 14, 14, 512)	2,359,808
MaxPooling2D	(None, 7, 7, 512)	0
Flatten	(None, 25088)	0
Dense	(None, 256)	6,422,784
Dense	(None, 128)	32,896
Dense	(None, 13)	1,677
Total params	**21,172,045**	
Trainable params	**21,172,045**	
Non-trainable params	**0**	

Figure 4: VGG16 Model Summary Before applying transfer learning

1.3.3 Transfer Learning

To adapt the VGG16 model to our task on the dataset, we used transfer learning. We froze the weights of all layers in the VGG16 model except for the last three fully connected layers. We then added a new output layer with 13 nodes corresponding to the 13 object categories in our dataset. We initialized the weights of the new output layer randomly and trained only the weights of the output layer while keeping the weights of the rest of the model fixed.

Layer (type)	Output Shape	Param #
conv2d_39 (Conv2D)	(None, 224, 224, 64)	1792
conv2d_40 (Conv2D)	(None, 224, 224, 64)	36928
max_pooling2d_13 (MaxPooling2D)	(None, 112, 112, 64)	0
conv2d_41 (Conv2D)	(None, 112, 112, 128)	73856
conv2d_42 (Conv2D)	(None, 112, 112, 128)	147584
max_pooling2d_14 (MaxPooling2D)	(None, 56, 56, 128)	0
conv2d_43 (Conv2D)	(None, 56, 56, 256)	295168
conv2d_44 (Conv2D)	(None, 56, 56, 256)	590080
conv2d_45 (Conv2D)	(None, 56, 56, 256)	590080
max_pooling2d_15 (MaxPooling2D)	(None, 28, 28, 256)	0
conv2d_46 (Conv2D)	(None, 28, 28, 512)	1180160
conv2d_47 (Conv2D)	(None, 28, 28, 512)	2359808
conv2d_48 (Conv2D)	(None, 28, 28, 512)	2359808
max_pooling2d_16 (MaxPooling2D)	(None, 14, 14, 512)	0
conv2d_49 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_50 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_51 (Conv2D)	(None, 14, 14, 512)	2359808
vgg16 (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 256)	6422784
fc2 (Dense)	(None, 128)	32896
output (Dense)	(None, 13)	1677
Total params:	**21,172,045**	
Trainable params:	**6,457,357**	
Non-trainable params:	**14,714,688**	

Figure 5: VGG16 Model Summary after applying transfer learning freezing last 3 layers

1.3.4 Training

We used the Adam optimizer with a learning rate of 0.001 to train the VGG16 model on our dataset. We used the categorical cross-entropy loss function and monitored the accuracy metric during training. To prevent overfitting, we used early stopping with a patience of 20 epochs and a minimum improvement in validation accuracy of 0. We trained the model for 100 epochs and used a batch size of 32.

1.4 Training Results

The results of this study demonstrated that the VGG16 model, trained on a subset of 13 categories from the TU-Berlin sketch dataset, achieved an accuracy of 98% on the test set. This suggests that the VGG16 model is capable of accurately classifying sketches from a limited set of categories with high precision. The use of the TU-Berlin sketch dataset proved advantageous due to its large number of high-quality sketches, which allowed for more robust training and testing of the model. Additionally, the limited computational power of the PC necessitated the use of a smaller subset of categories, which did not significantly impact the accuracy of the model. Overall, these results suggest that the VGG16 model is a powerful tool for sketch classification, particularly when used with high-quality datasets such as TU-Berlin.

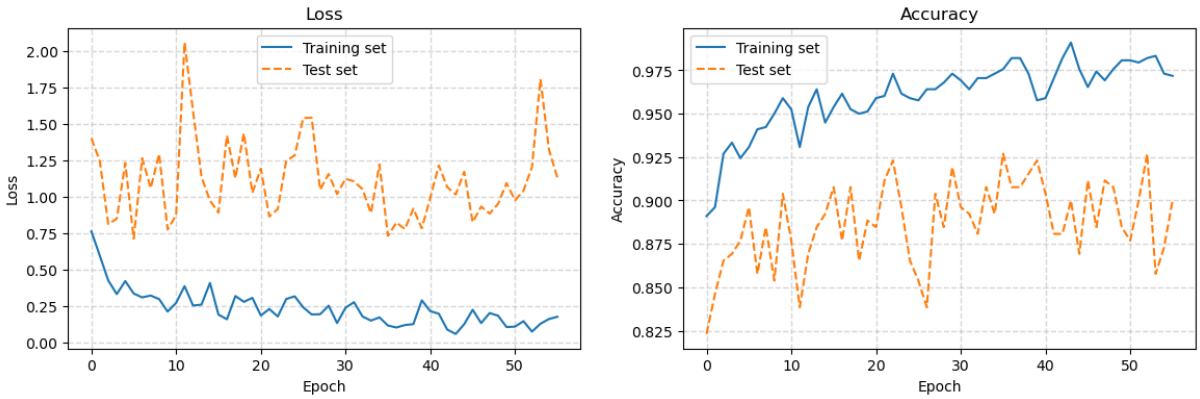

Figure 6: Loss and Accuracy Graphs of VGG16 Model while training on the dataset.

1.5 Demo Exhibition 1

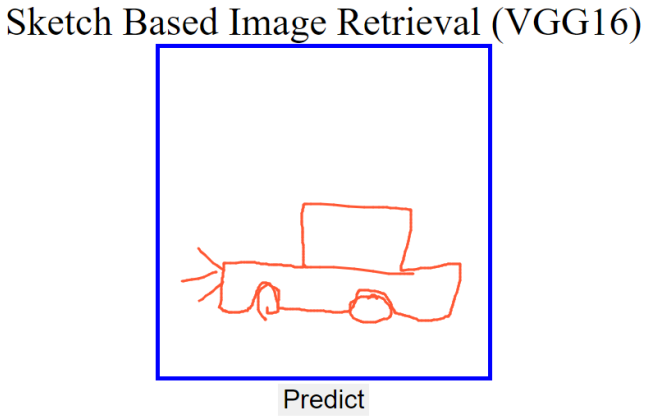

Figure 7: Input sketch of a car to the VGG16 Model.

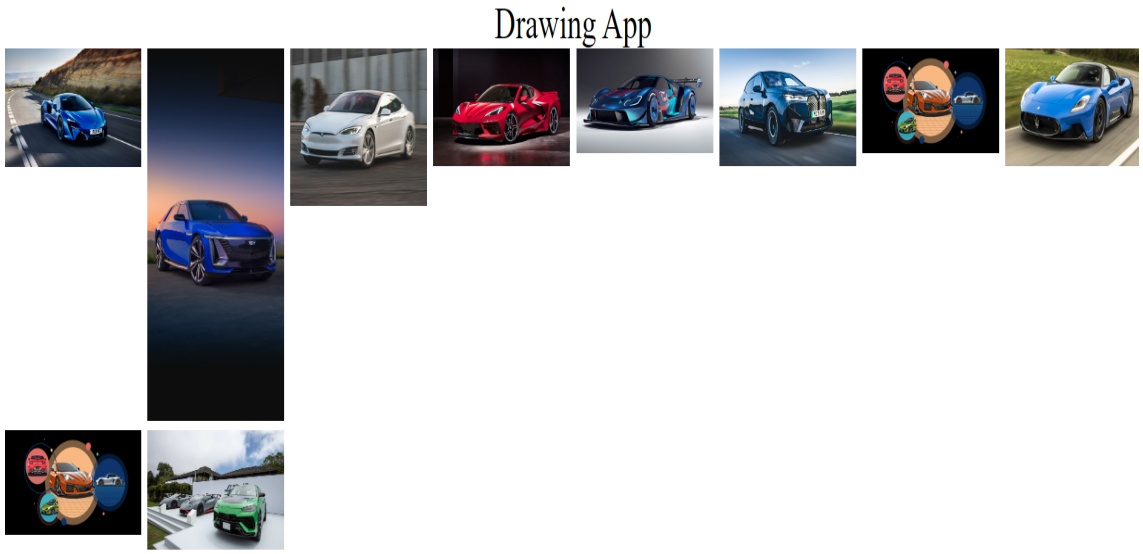

Figure 8: Results after the VGG16 Model Classified the sketch.

1.6 Demo Exhibition 2

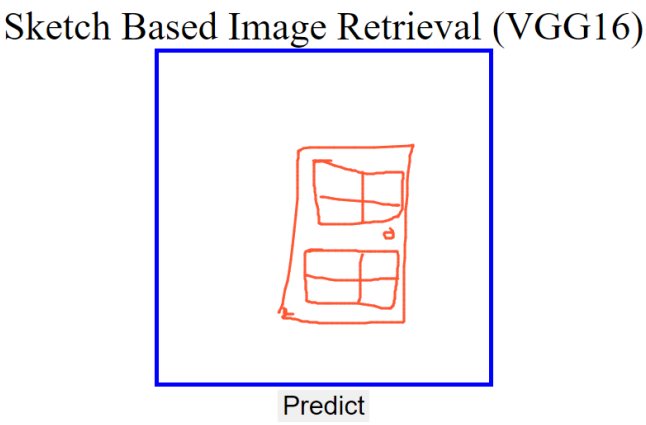

Figure 9: Input sketch of a door to the VGG16 Model.

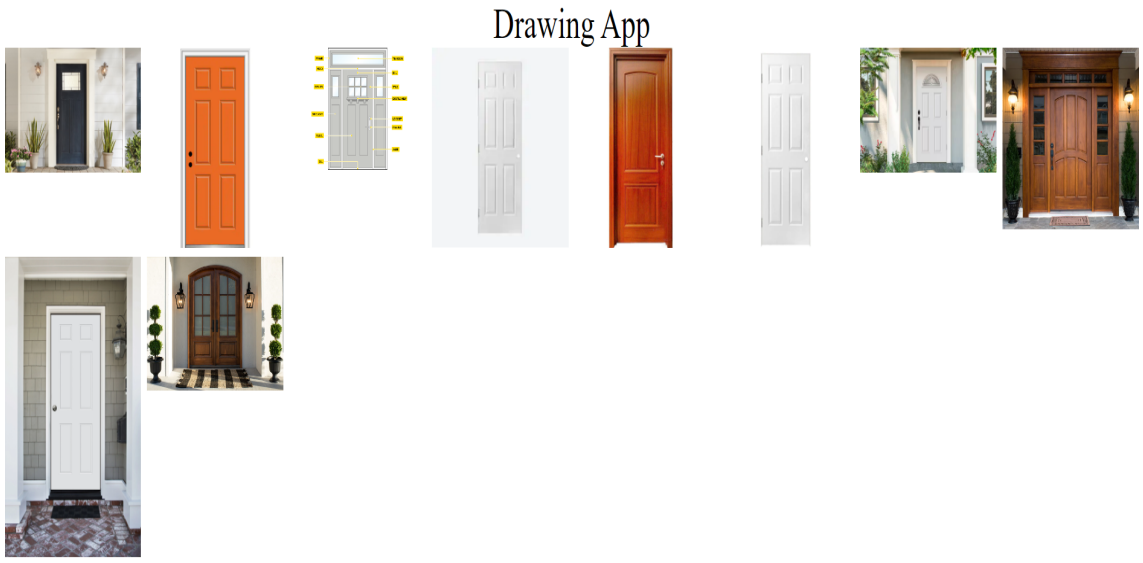

Figure 10: Results after the VGG16 Model Classified the sketch.

References

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[2] W. Wang, J. Yan, J. Yan, and S. Xiang, "Remote sensing image retrieval using deep convolutional neural network," *Neurocomputing*, vol. 380, pp. 110–118, 2020.