



CONVERSE®

Brand Classification Project

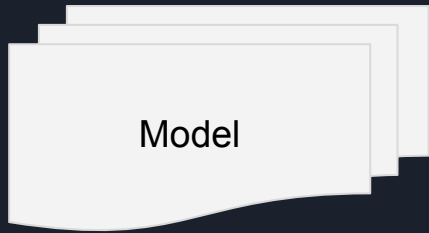
by Oleg Budachov

1. Primary goal of the project

The primary goal of this project is to categorize images of shoes into three distinct brands: Nike, Adidas, and Converse by employing transfer learning with the InceptionV3 model, a deep convolutional neural network (CNN) architecture developed by Google's research team, which is primarily designed for image classification and object recognition tasks.



Images of shoes



2. Data overview

The dataset is structured into two folders, one for test data and the other for training data, with a test-train-split ratio of 0.14. The test dataset comprises 114 images, while the training dataset consists of 711 images. All images are in RGB color format and have a resolution of 240x240 pixels. Within both folders, there are three distinct classes: Adidas, Converse, and Nike.

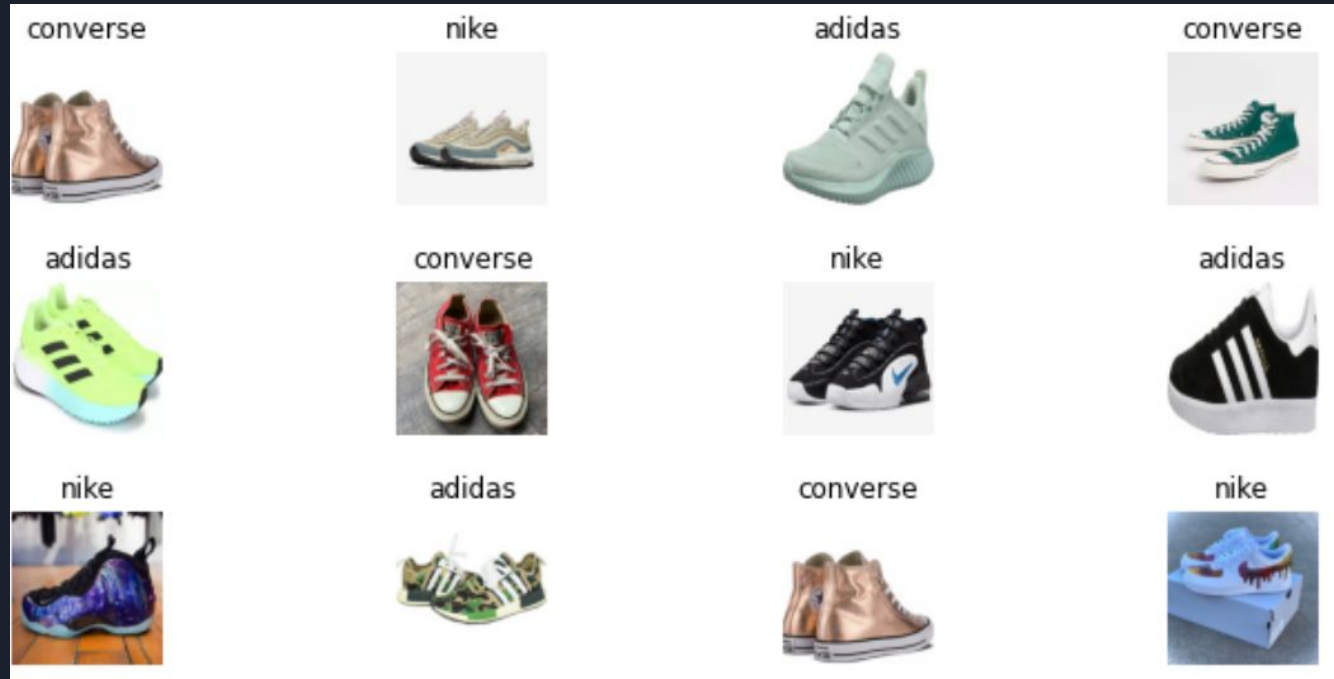
This dataset was acquired by downloading images from Google Images in .webp format than they were converted to .jpg format. Subsequently, the images were randomly shuffled and resized to ensure a consistent resolution of 240x240 pixels for all samples.



3. Data preprocessing and EDA

Let's confirm that our data is balanced across all three classes:

Let's visualize a random sample of images from our dataset, which can help us assess the quality of the data, check for any potential issues, and get a sense of the visual characteristics of our dataset:



```

datagenerator = {
    "train": ImageDataGenerator(horizontal_flip=True,
                                vertical_flip=True,
                                rescale=1. / 255,
                                validation_split=0.1,
                                shear_range=0.1,
                                zoom_range=0.1,
                                width_shift_range=0.1,
                                height_shift_range=0.1,
                                rotation_range=30,
                                ).flow_from_directory(directory=base_dir,
                                                         target_size=(300, 300),
                                                         subset='training',
                                                         ),
    "valid": ImageDataGenerator(rescale=1. / 255,
                                validation_split=0.1,
                                ).flow_from_directory(directory=base_dir,
                                                         target_size=(300, 300),
                                                         subset='validation',
                                                         ),
}

```

Create a generator for images in the specified subset

```
image_generator = datagenerator["train"]
```

Generate and visualize augmented images

```
num_samples = 5 # Adjust the number of augmented images to display
plt.figure(figsize=(12, 6))
```

```
for i in range(num_samples):
```

```
    augmented_image, class_label = next(image_generator) # Generate
    image = augmented_image[0] # Extract the image data from the generator
    class_index = np.argmax(class_label[0]) # Convert one-hot encoding to class index
```

Display the augmented image

```
    plt.subplot(1, num_samples, i + 1)
    plt.imshow(image)
    plt.title(f'Class Index: {class_index}')
    plt.axis('off')
```

```
plt.tight_layout()
```

```
plt.show()
```

Class Index: 2



Class Index: 1



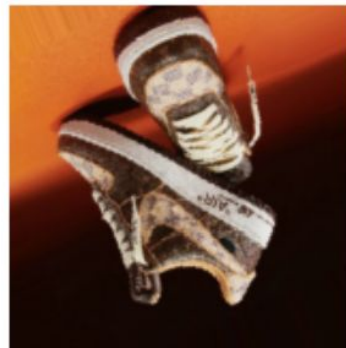
Class Index: 2



Class Index: 2



Class Index: 2



4. Model building: Initializing InceptionV3

```
# Initializing InceptionV3 (pretrained) model with input image shape as (300, 300, 3)  
base_model = InceptionV3(weights=None, include_top=False, input_shape=(300, 300, 3))
```

```
# Load Weights for the InceptionV3 Model
```

```
base_model.load_weights('../input/inceptionv3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5')
```

```
# Setting the Training of all layers of InceptionV3 model to false
```

```
base_model.trainable = False
```

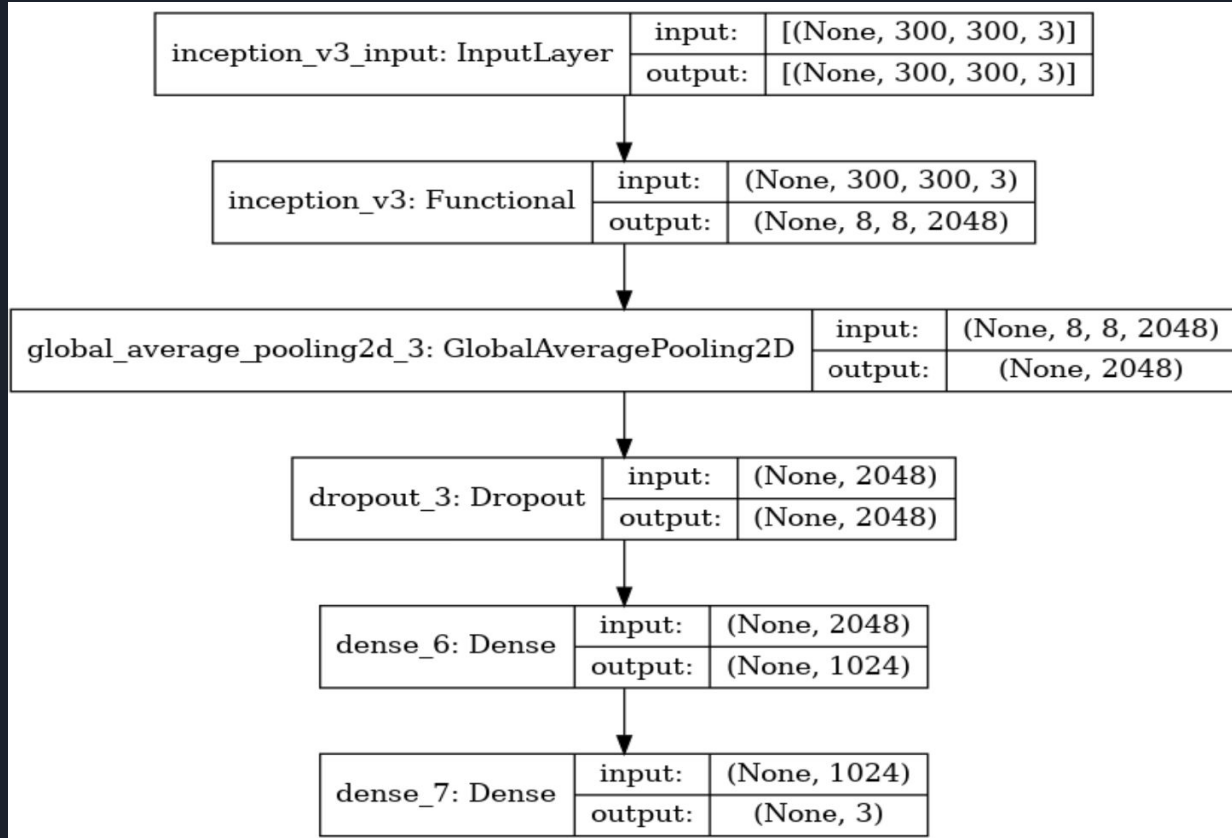
```
# Adding some more layers at the end of the Model as per our requirement
```

```
model = Sequential([  
    base_model,  
    GlobalAveragePooling2D(),  
    Dropout(0.15),  
    Dense(1024, activation='relu'),  
    Dense(3, activation='softmax')  
])
```

```
# Using the Adam Optimizer to set the learning rate of our final model
```


```
o = optimizers.Adam(learning_rate=0.0001)
```

View model summary & plot





5. Model training



```
# File Path to store the trained models
filepath = "./model_{epoch:02d}-{val_accuracy:.2f}.h5"

# Using the ModelCheckpoint function to train and store all the best models
checkpoint1 = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')

callbacks_list = [checkpoint1]
# Training the Model
history = model.fit_generator(generator=train_generator, epochs=epochs, steps_per_epoch=steps_per_epoch,
                             validation_data=valid_generator, validation_steps=validation_steps,
                             callbacks=callbacks_list)
```

Epoch 00007: val_accuracy did not improve from 0.75000

Epoch 8/10

20/20 [=====] - 14s 733ms/step - loss: 0.5954 - accuracy: 0.7426 - val_loss: 0.6888 - val_accuracy: 0.7344

Epoch 00008: val_accuracy did not improve from 0.75000

Epoch 9/10

20/20 [=====] - 15s 734ms/step - loss: 0.5821 - accuracy: 0.7639 - val_loss: 0.6144 - val_accuracy: 0.8125

Epoch 00009: val_accuracy improved from 0.75000 to 0.81250, saving model to ./model_09-0.81.h5

Epoch 10/10

20/20 [=====] - 14s 691ms/step - loss: 0.6205 - accuracy: 0.7557 - val_loss: 0.6498 - val_accuracy: 0.7500

Epoch 00010: val_accuracy did not improve from 0.81250

6. Model Evaluation and Tuning

Epoch 00006: val_accuracy did not improve from 0.71875

Epoch 7/10

20/20 [=====] - 14s 692ms/step - loss: 0.5720 - accuracy: 0.7672 - val_loss: 1.0237 - val_accuracy: 0.5469

Epoch 9/10

20/20 [=====] - 15s 734ms/step - loss: 0.5821 - accuracy: 0.7639 - val_loss: 0.6144 - val_accuracy: 0.8125

Epoch 00008: val_accuracy improved from 0.71875 to 0.73438, saving model to ./model_08-0.73.h5

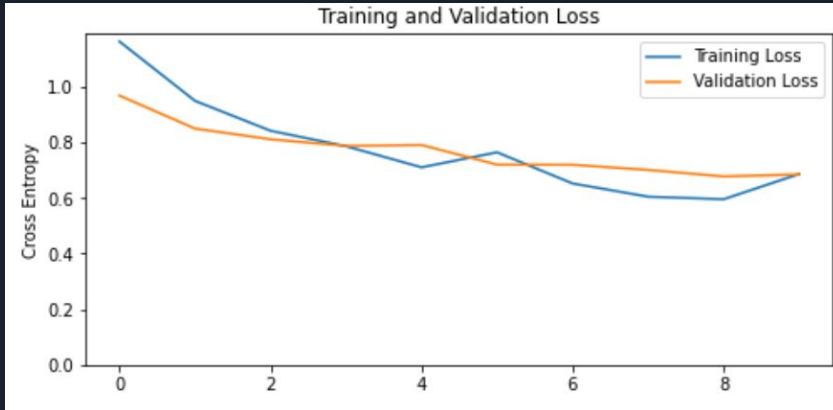
Epoch 9/10

20/20 [=====] - 14s 699ms/step - loss: 0.5694 - accuracy: 0.7541 - val_loss: 0.8090 - val_accuracy: 0.6562

Epoch 8/10

20/20 [=====] - 15s 736ms/step - loss: 0.6098 - accuracy: 0.7475 - val_loss: 0.7275 - val_accuracy: 0.7344

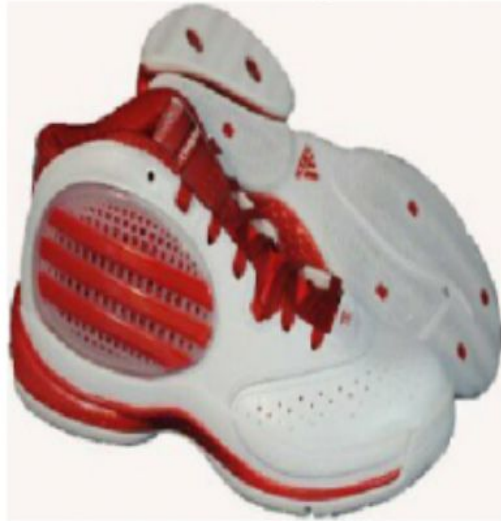
Epoch 00010: val_accuracy did not improve from 0.73438



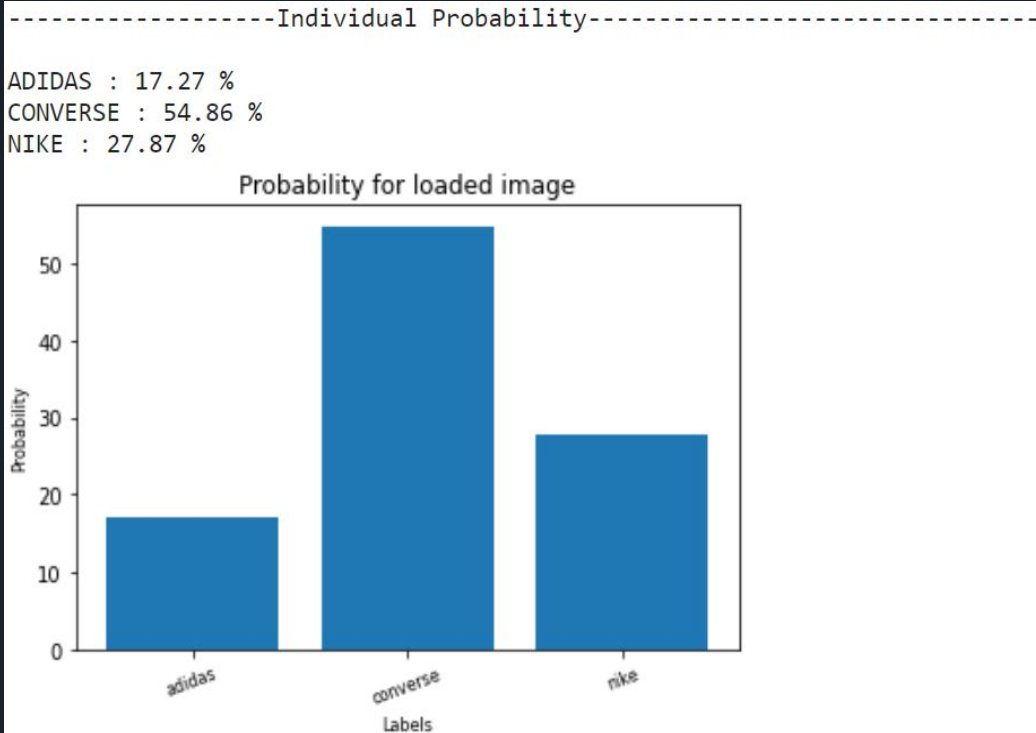
7. Model Testing With Graphs

3/3 [=====] - 1s 262ms/step - loss: 0.6689 - accuracy: 0.7826
test accuracy : 0.782608687877655

Loaded Image



Maximum Probability: 0.54862535
Classified: converse





8. Conclusion and further Improvements

- **Task:** Image classification for shoe brands (Adidas, Converse, Nike).
- **Model:** Deep learning based on InceptionV3 architecture.
- **Results:** Achieved 78% test accuracy, demonstrating generalization.

Practical Applicability:

- Successful classification of brand logos.
- Model's relevance in real-world scenarios.

Future Improvements:

- Data Augmentation
- Fine-Tuning
- Hyperparameter Tuning
- Regularization
- Ensemble Learning
- Advanced Metrics
- Interpretability
- Continuous Monitoring



CONVERSE®