

**Team Members:**

**I A Thangamma - 21BSR18043**

**Jahanvi Agrawal - 21BSR18029**

**Yaseen Ali Khan - 21BSR18055**

**Romit Dutta - 21BSR18031**

**Course: B.Sc. (Hons) Data Science & Analytics**

**Semester: V**

**PROJECT REPORT**

**DEEP LEARNING ACTIVITY 2**

**DEEP LEARNING ACTIVITY 2**

**PROJECT REPORT**

**JAIN UNIVERSITY**

**School of Sciences**

**Course: B.Sc. (Hons) Data Science & Analytics**

**Semester: V**

**Team Members:**

**I A Thangamma - 21BSR18043**

**Jahanvi Agrawal - 21BSR18029**

**Yaseen Ali Khan - 21BSR18055**

**Romit Dutta - 21BSR18031**

**INTRODUCTION:**

**What is CNN?**

CNN stands for Convolutional Neural Network. It is a deep-learning neural network architecture that is particularly well-suited for image recognition and classification tasks. CNNs are inspired by the structure and function of the human visual cortex, and they learn to detect patterns in images by applying a series of convolutional filters to the input image.

A CNN typically consists of the three main layers:

**Convolutional layers**: These layers apply convolutional filters to the input image. The filters are small matrices of weights that are learned during training. Each filter is applied to the input image at different positions, and the output of each filter is a feature map.

**Pooling layers**: These layers downsample the feature maps from the convolutional layers. This helps to reduce the size of the network and makes it more computationally efficient.

**Fully connected layers**: These layers are similar to the layers in a traditional neural network. They are used to classify the input image based on the features extracted by the convolutional and pooling layers.

CNNs have achieved state-of-the-art results on a wide range of image recognition tasks, including object classification, object detection, and image segmentation. They are also used in other domains, such as natural language processing and speech recognition.

**OVERVIEW OF THE CODE:**

The following code which was done in activity 1 implements the Convolutional Neural Network (CNN) model, which is used for image classification using TensorFlow and Keras.

It is a comprehensive pipeline for training an image classification mode, evaluating its performance, saving the model, and creating a user-friendly interface for making predictions using Gradio.

It contains the following steps:

* Data Collection
* Data Preprocessing
* Model Building
* Model Training
* Model Preservation
* Hyperparameter Tuning
* Gardio Interface

**DATA COLLECTION:**

* The dataset was collected from the Internet and manually, where the dataset was divided into training, testing, and validation.
* The dataset also was classified into three classes – Adidas, Nike, and Converse, the dataset has been uploaded into Google Drive in different folders, where the train has 70%, the test has 20% and validation has 10%.

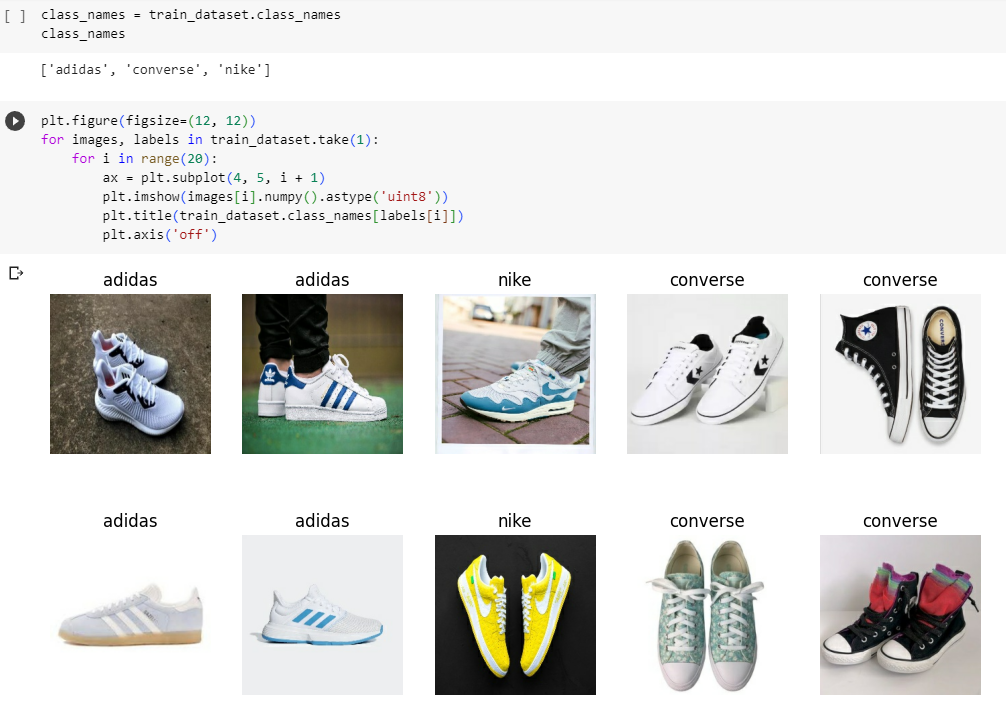
Link of the dataset: <https://drive.google.com/drive/folders/17IJ0nGIQ9CHzZBWLHL5WJV0fmjCMQK3a?usp=sharing>.

**DATA PREPARATION:**

* The code is started by mounting Google Drive and defining the parameters batch size and image size for the image data.
* By using TensorFlow’s ‘*image\_dataset\_from\_directory*’ function we are creating training, validation, and test datasets from image directories, which is essential for training and model evaluation.



* We are using ‘*class\_names’* code to extract class names from the training dataset, which will be used for label interpretation later.
* Then visualization of a sample of images from the training dataset is done using Matplotlib to monitor models in real time and to identify any issues are there in the data.

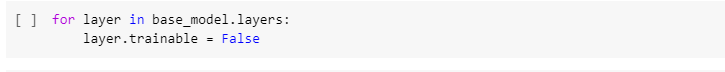


**MODEL BUILDING:**

* The model is built upon the Xception architecture, a pre-trained model on the ImageNet dataset, which serves as a feature extractor.
* This is the base model which will be used for transfer learning.[As the dataset is small, we are going to use transfer learning]
* Input images are resized to 224x224 pixels and pre-processed using the Xceptionspecific preprocessing function.
* Data augmentation layers (horizontal flip, rotation, and zoom) are added to the input pipeline to increase the model's robustness.
* The architecture is built upon the base model, including a GlobalAveragePooling layer, a dropout layer, and a fully connected output layer with softmax activation for classification, where the GlobalAveragePooling layer and dropout layer are used for regularization of the model and in the fully connected output layer (which is the final dense layer) with softmax activation function is used for multiclass classification.



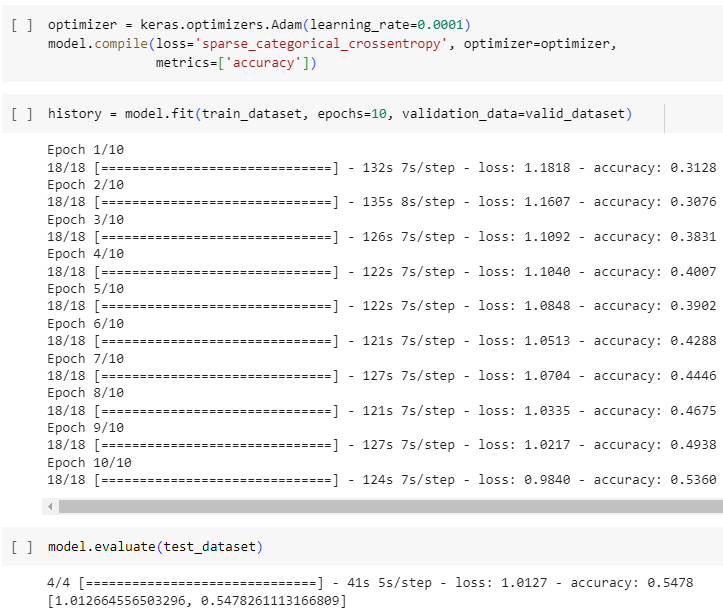
* The Xception base model's layers are frozen to retain its pretrained knowledge.



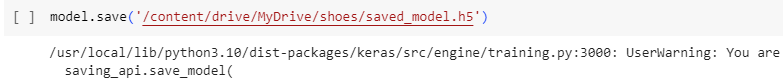
* Transfer learning allows the model to leverage the feature extraction capabilities of Xception while finetuning the top layers for the specific task of image classification.
* This approach is useful for improving training efficiency and achieving good performance with a smaller dataset.

**MODEL TRAINING:**

* The model has complied with the Adam optimizer and sparse categorical cross-entropy loss.
* The model is trained using the training dataset with 10epocs and validation data is specified.
* Training of the model is done by using metrics such as loss and accuracy.
* After training, model evaluation is done to evaluate the model’s performance on the test dataset.



**MODEL PRESERVATION:**

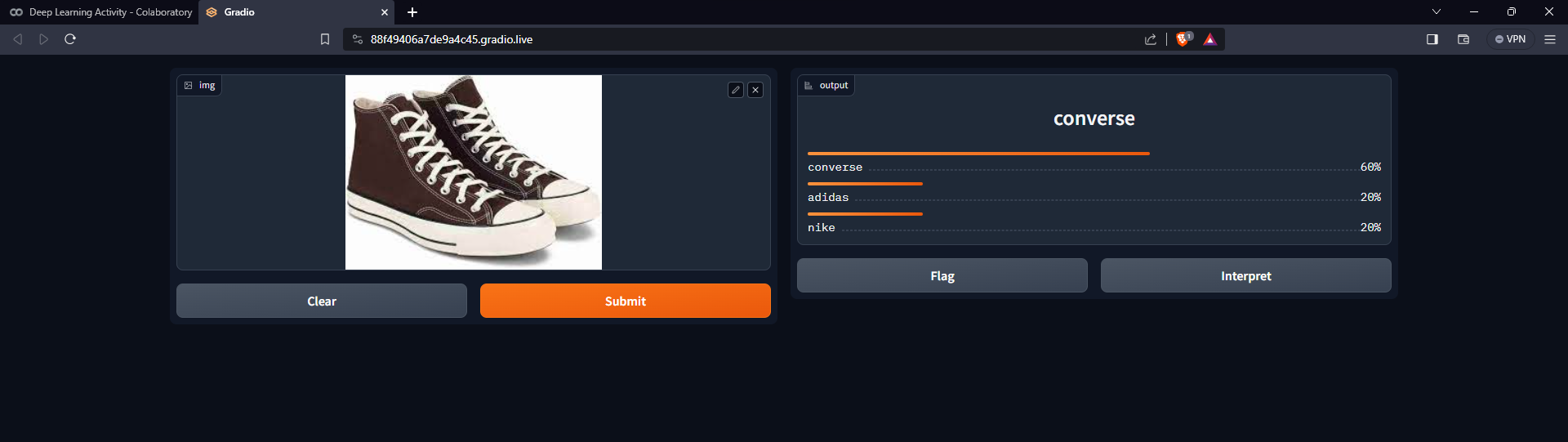
* After training the model it is saved to a file ‘*/content/drive/MyDrive/shoes/saved\_model.h5*’ for future use, such as to ensure the continued accuracy and performance of the models, to facilitate collaboration and knowledge sharing, etc.

**HYPERPARAMETER TUNING:**

* In order to achieve higher accuracy, the weights of the pre-trained layers will be unfrozen.
* The learning rate will be lowered to avoid damaging the pre-trained weights.
* To avoid overfitting and speed up training (if the quality of the model won't be increasing) early stopping will be used.
* Since early stopping is used, the number of epochs will be set to 20.
* The model is trained again and evaluation is performed again.



**GRADIO INTERFACE:**

* The ‘*predict\_image’* function is defined, which takes an image as input, reshapes it, and uses the trained model to make predictions. It returns the top 3 class predictions with their probabilities.
* The Gradio library is used to create a simple web interface for making predictions with the trained model.
* An image input component and label output component are defined for the Gradio interface. The ‘*predict\_image’* function is used as the prediction function.
* The Gradio interface is launched for real-time predictions.
* Users can upload an image, and the model provides the top three predicted class labels along with their probabilities.

**CONCLUSION:**

The provided code demonstrates a complete pipeline for training, evaluating, saving, and deploying an image classification model using TensorFlow/Keras, Gradio for creating an interactive interface, and adjustable hyperparameters for model tuning.

**Project Link:**

https://github.com/IAThangamma/Deep-Learning-Gradio-

The CNN model created serves as a practical example of image classification using deep learning techniques and also showcases the use of transfer learning to leverage the pre-trained models for better performance.

By this, we can say that our model can classify different types of shoes and thus we have achieved this through Convolutional Neural Network (CNN).