Training On Deep Learning

BY

Ayushman Dutta

BTECH/10062/20

Electrical and Electronics Engineering
BIRLA INSTITUTE OF TECHNOLOGY
MESRA-835215, RANCHI

APPROVAL OF THE MENTOR

Recommended that the summer training entitled **"Training on Deep Learning"** presented by **Ayushman Dutta** under my supervision and guidance be accepted as fulfilling the partial requirements for the award of Degree of **Bachelor of Technology in Electrical and Electronics Engineering.** To the best of my knowledge, the content of this training did not form a basis for the award of any previous degree to anyone else.

Date: 02/08/2023

Mentor
Abhishek Bansal

INTERNSHALA

DECLARATION CERTIFICATE

I certify that

a) The work contained in the report is original and has been done by myself under the general supervision of my supervisor.

b) The work has not been submitted to any other Institute for any other degree or diploma.

c) I have followed the guidelines provided by the Institute in writing the thesis.

d) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e) Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

f) Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Ayushman Dutta

BTECH/10062/20

CERTIFICATE OF APPROVAL

This is to certify that the work embodied in this training entitled **"Training on Deep Learning"**, is carried out by **Ayushman Dutta (BTECH/10062/20)** has been approved for the degree of Electrical and Electronics Engineering of Birla Institute of Technology, Mesra, Ranchi.

Date:

Place:

Head of Department
Dept. Of EEE

Examiner
Dept. Of EEE

ABSTRACT

This summer training report documents my immersive experience in Deep Learning, where I worked on diverse projects to enhance my skills and understanding in this dynamic field. The projects included fashion classification using the FashionMNIST dataset, handwritten digit classification with sequential models, movie collection prediction utilizing sequential models, animal image classification with sequential models and data generators, house price prediction using sequential models, and face mask detection through VGG19 transfer learning.

In the fashion classifier project, I developed a robust clothing item classifier, employing convolutional neural networks (CNNs) on the FashionMNIST dataset. The handwritten digit classification project involved sequential models to accurately recognize handwritten digits in the MNIST dataset.

Next, I ventured into regression systems in the movie collection prediction project, building a movie box office collection model using sequential models. In the animal image classification project, I curated a custom dataset and leveraged sequential models with data generators to classify various animal species.

For house price prediction, I employed sequential models for regression to forecast property prices based on various features, emphasizing feature engineering and data normalization. Lastly, I utilized VGG19 transfer learning in the face mask detection project, developing an efficient system for identifying individuals wearing face masks.Throughout the training, I encountered diverse challenges and expanded my expertise in Deep Learning techniques, preparing me for future endeavours in the fields of artificial intelligence and data science.

ACKNOWLEDGEMENT

I would like to express my profound gratitude to my training guide, Abhishek Bansal for his guidance and support during my training work. I benefited greatly by working under his guidance. It was his effort for which I am able to develop a detailed insight on this subject and special interest to study further.

I would like to thank my college and professors for providing me the opportunity to work at Internshala. I must express my very profound gratitude to my parents and friends for providing me with unfailing support and continuous encouragement throughout the years of my study. This accomplishment would not have been possible without them.

Thank you.

DATE:	Name of the Student	(Roll Number)
02/08/23	Ayushman Dutta	BTECH/10062/20

CONTENTS

Introduction to Deep Learning and Key Terminologies:

Deep Learning is a subset of machine learning that focuses on training artificial neural networks to perform complex tasks by mimicking the human brain's learning process. It has gained immense popularity due to its ability to tackle challenging problems across various domains, including computer vision, natural language processing, speech recognition, and more.

Terminologies in Deep Learning:

Neural Network: A collection of interconnected artificial neurons organized in layers. It is the fundamental building block of Deep Learning models.

Artificial Neuron (Perceptron): A computational unit that takes inputs, applies weights and biases, and produces an output using an activation function.

Activation Function: A non-linear function that introduces non-linearity into the neural network, allowing it to learn complex relationships in the data.

Layers: Neural networks consist of multiple layers, including an input layer, one or more hidden layers, and an output layer.

Hidden Layers: Layers between the input and output layers that process data and learn patterns from the data.

Weights and Biases: Parameters in neural networks that are adjusted during training to optimize the model's performance.

Model: A neural network architecture or structure designed to solve a specific problem. It comprises interconnected layers and parameters that are learned during training.

Keras: An open-source Deep Learning library that provides a user-friendly API for building and training neural networks. It is built on top of TensorFlow and is widely used for its simplicity and flexibility.

Filters (Kernels): Small windows used in convolutional layers to scan input data and extract features. They play a crucial role in identifying patterns in images and other spatial data.

Optimizer: An algorithm used to update the weights and biases of a neural network during training to minimize the loss function. Common optimizers include Stochastic Gradient Descent (SGD), Adam, and RMSprop.

Metrics: Performance measures used to evaluate the model's performance, such as accuracy, precision, recall, and F1-score.

Loss Function (Cost Function): A measure of the difference between the predicted output and the actual output used to assess how well the model is performing.

Backpropagation: An optimization algorithm used to adjust the weights and biases of a neural network based on the loss function's gradient to minimize the error.

Gradient Descent: An optimization technique that iteratively updates the weights and biases in the direction that reduces the loss function.

Overfitting: When a model becomes too specialized on the training data and performs poorly on new, unseen data.

Underfitting: When a model is too simplistic to capture the underlying patterns in the data, resulting in poor performance on both training and test data.

Batch Size: The number of training examples used in one iteration of the gradient descent optimization process.

Epoch: One pass through the entire training dataset during the training process.

Learning Rate: A hyperparameter that determines the step size at which the model's weights and biases are updated during gradient descent.

Transfer Learning: A technique that involves using a pre-trained model as a starting point for a new, related task, allowing faster training with limited data.

Deep Learning, with its powerful algorithms and versatile tools like Keras, has revolutionized artificial intelligence, making it a driving force behind technological advancements and applications in various industries. Understanding these key terminologies is essential for effectively exploring and leveraging the potential of Deep Learning in solving real-world problems.

Project 1: Handwritten Digit Classifier

Introduction:

In this project, I built a handwritten digit classifier using the MNIST dataset. The goal was to accurately identify handwritten digits from 0 to 9 using Deep Learning techniques. I implemented a Convolutional Neural Network (CNN) model with multiple layers to achieve high accuracy in digit recognition. This report provides an overview of the model architecture, training process, and evaluation metrics, along with their interpretations.

Model Architecture:

The model was constructed using the Keras Sequential API. It began with a Conv2D layer with 32 filters, a kernel size of (3,3), and ReLU activation to detect patterns in the input images (28x28 pixels with one color channel). The MaxPooling2D layer with a pool size of (2,2) was used to downsample the feature maps and reduce computational complexity.

Afterwards, the Flatten layer was introduced to transform the pooled feature maps into a 1D vector to prepare for fully connected layers. Two Dense layers with 200 and 100 neurons, respectively, were added with ReLU activation to introduce non-linearity and extract relevant features. The final Dense layer with 10 neurons used the softmax activation function to provide probabilities for each digit class (0 to 9).

Model Compilation:

The model was compiled with the "sparse_categorical_crossentropy" loss function, as we are dealing with integer labels (0 to 9) rather than one-hot encoded labels. The "sgd" (Stochastic Gradient Descent) optimizer was chosen for optimization, which helps in iteratively adjusting the model's parameters to minimize the loss. The performance of the model was evaluated based on the "accuracy" metric, which measures the percentage of correctly predicted digits.

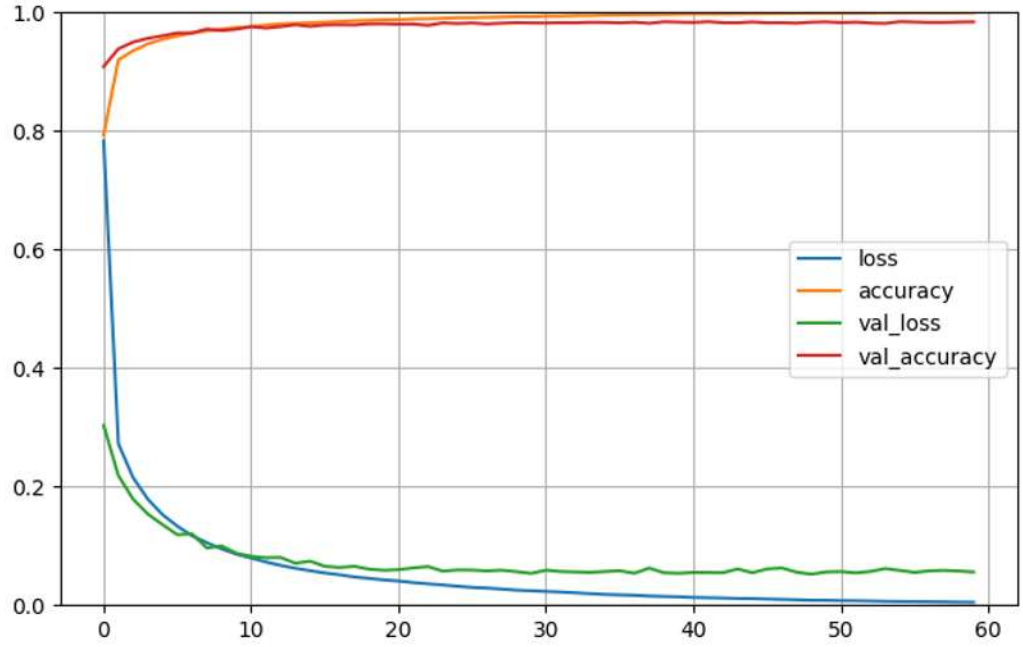
Training and Results:

The model was trained on the MNIST dataset, consisting of 60,000 training samples and 10,000 test samples. The training process involved iterative optimization of the model's weights over epochs. The training was performed on a batch of images at each iteration, where each batch size was set to a default value.

The model achieved remarkable results with a loss of 0.0562 and an accuracy of 98.35% on the test dataset. The high accuracy indicates that the model has effectively learned to recognize handwritten digits, making it a reliable classifier for practical applications.

Conclusion: This project demonstrates the successful implementation of a handwritten digit classifier using a Convolutional Neural Network. The model achieved impressive accuracy in recognizing digits from the MNIST dataset. The Conv2D layers helped capture important patterns and features from the images, while the Dense layers performed the final classification task. This project showcases the power and versatility of Deep Learning in solving complex image recognition problems and opens the door for further exploration and applications in the field of computer vision.

Accuracy and Loss Graph [X-axis – Epochs, Y-axis – Loss and Accuracy]



Project 2: Fashion Classifier

Introduction:

In this project, I developed a fashion classifier using the Fashion MNIST dataset. The primary objective was to accurately classify various fashion items, such as clothing, shoes, and accessories. This report outlines the model architecture, training process, and evaluation metrics to provide insights into the classifier's performance.

Model Architecture:

The model was constructed using the Keras Sequential API, which allows for a simple linear stack of layers. The initial layer, Flatten, transformed the 28x28 pixel images into a 1D array of 784 values, preparing the data for the subsequent fully connected layers.

Three Dense layers were introduced with 392, 196, and 98 neurons, respectively. Each hidden layer used the ReLU activation function, enabling the model to learn complex patterns and features from the flattened input data.

The final Dense layer with 10 neurons used the softmax activation function, providing probabilities for each fashion class (0 to 9). The class with the highest probability indicated the predicted fashion item.

Model Compilation:

The model was compiled using the "sparse_categorical_crossentropy" loss function, which is suitable for integer labels (0 to 9) without requiring one-hot encoding. For optimization, the stochastic gradient descent (SGD) optimizer was employed. The optimizer iteratively adjusted the model's weights and biases during training to minimize the loss and improve accuracy.

The performance of the model was evaluated using the "accuracy" metric, which measures the percentage of correctly predicted fashion classes from the test dataset.

Training and Results: The model was trained on the Fashion MNIST dataset, consisting of 60,000 training samples and 10,000 test samples. Throughout training, the model learned from the training data, iteratively adjusting its parameters to minimize the loss.

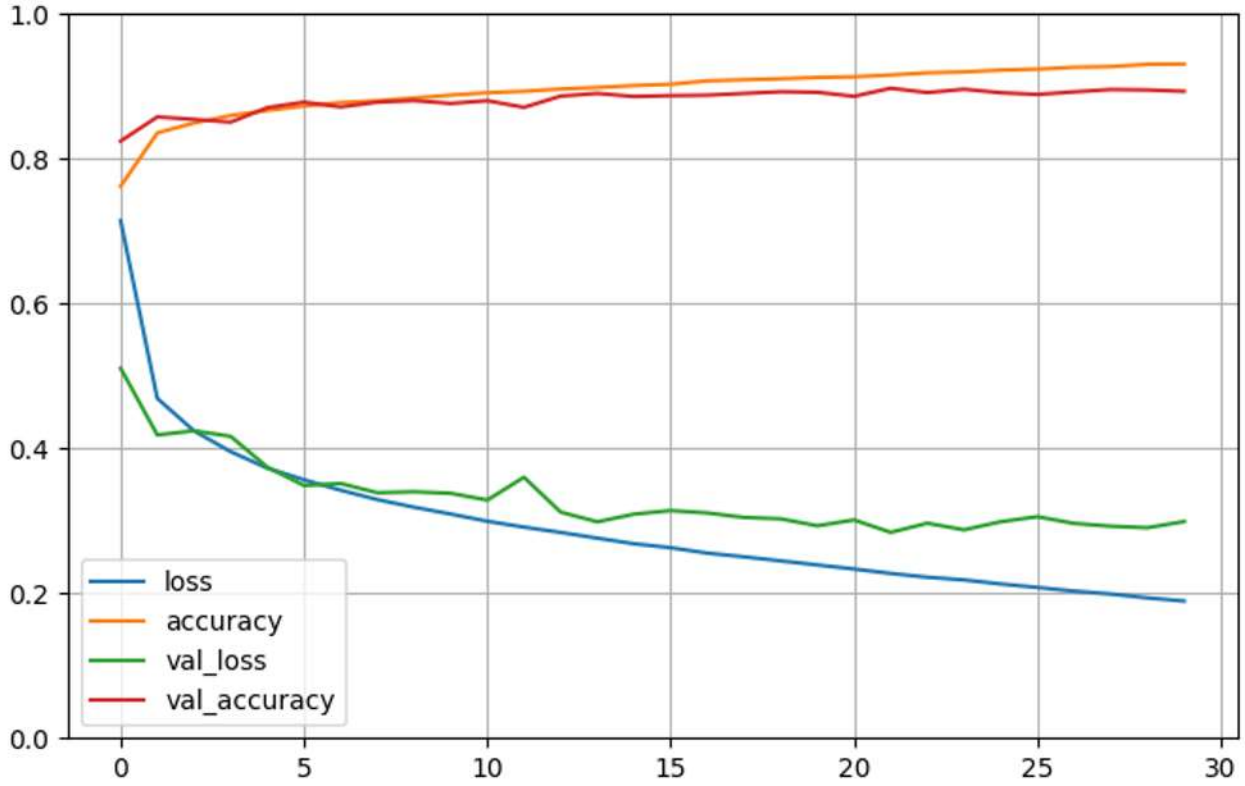
The training process yielded a satisfactory result, with a final loss of 0.3344 and an accuracy of 88.41% on the test dataset. The accuracy demonstrates that the model has learned to classify fashion items with reasonable precision.

Conclusion:

The successful development of the fashion classifier using a Deep Learning model on the Fashion MNIST dataset showcases the effectiveness of neural networks in image classification tasks. The model's architecture, activation functions, and optimization methods contributed to its decent accuracy. This project highlights the potential of Deep Learning in practical applications, allowing fashion retailers and e-commerce platforms to employ such classifiers for automated fashion item

categorization. Further fine-tuning and experimentation could potentially enhance the model's performance, making it more robust and reliable for real-world usage.

Accuracy and Loss Graph [X-axis – Epochs, Y-axis – Loss and Accuracy]



Project 3: Movie Box Office Collection Predictor

Introduction:

In this project, I developed a movie box office collection predictor using a machine learning model. The goal was to build a regression model that can predict the box office collection of movies based on certain features. This report outlines the data preprocessing steps, model training, evaluation metrics, and suggestions for improving accuracy.

Data Preprocessing:

The data preprocessing involved using the StandardScaler from the sklearn.preprocessing module. Standardization was applied to scale the features in the training, testing, and validation datasets to have zero mean and unit variance. This step helps in improving the convergence rate during training and ensures that all features contribute equally to the predictions.

Model Architecture:

The movie collection predictor model is a simple feedforward neural network with two hidden layers, each having 30 neurons and the Rectified Linear Unit (ReLU) activation function. It takes 19 input features and outputs a single value, representing the predicted movie collection. The model has a total of 1,561 trainable parameters, making it relatively lightweight and suitable for this task. However, its accuracy and performance depend on the quality and size of the dataset used for training and the hyperparameters selected during model optimization. Further experimentation and fine-tuning are recommended to improve the model's accuracy and ensure reliable predictions for movie collection values. It seems to use a Mean Squared Error (MSE) loss function, which is suitable for regression tasks. The model is optimized using Stochastic Gradient Descent (SGD) with a learning rate of 1e-3.

Evaluation Metrics:

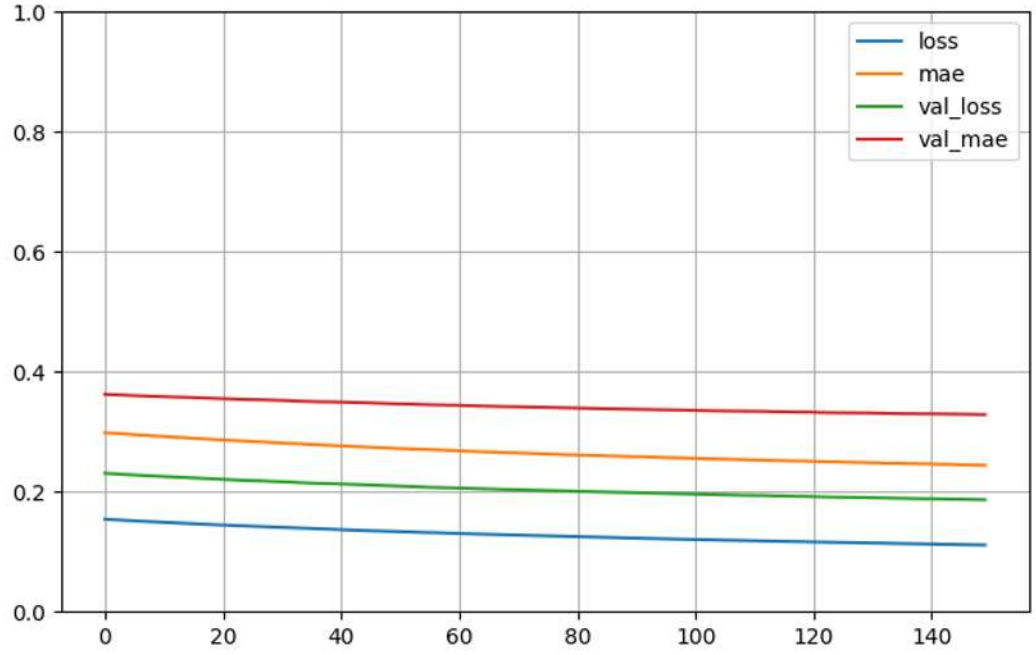
The model was evaluated using the Mean Squared Error (MSE) loss and Mean Absolute Error (MAE) metrics. The MSE measures the average squared difference between predicted and actual values, while MAE calculates the average absolute difference between predictions and ground truth values. Lower values for both metrics indicate better model performance.

Model Training and Results:

The model was trained for 150 epochs on the training data (xtrain and ytrain). The validation dataset (xvalid and yvalid) was used to monitor the model's performance during training.

After training, the model achieved a validation loss of 0.1904 and a Mean Absolute Error (MAE) of 0.3069, which suggests reasonably accurate predictions.

MAE and Loss Graph [X-axis – Epochs, Y-axis – Loss and MAE]



Suggestions for Improving Accuracy:

To further improve the accuracy of the movie box office collection predictor, consider the following steps:

Experiment with Model Architectures:

Try different regression model architectures, such as Gradient Boosting, Random Forest, or Neural Networks. Different models may capture different patterns in the data and lead to improved accuracy.

Hyperparameter Tuning:

Fine-tune the hyperparameters of the chosen regression model, including learning rate, batch size, and number of epochs. This can significantly impact model performance.

Feature Engineering:

Explore additional relevant features related to movies, such as genre, cast, director, release date, marketing budget, etc. Feature engineering can provide more valuable information to the model and enhance predictions.

Handling Outliers and Missing Data:

Address any outliers or missing data in the dataset appropriately. Outliers can distort predictions, and missing data can negatively impact model performance.

Ensemble Methods:

Consider using ensemble techniques like Bagging or Boosting, which combine multiple models to improve predictive accuracy.

Cross-Validation:

Implement k-fold cross-validation to get a more reliable estimate of the model's performance and avoid overfitting.

Regularization:

Apply regularization techniques like L1 or L2 regularization to prevent overfitting and improve generalization.

Project 4: House Price Predictor

Introduction:

In this project, a regression model was developed to predict house prices based on various features. Two different model architectures were explored, and the impact of using callbacks during training was investigated. This report outlines the data preprocessing steps, model architectures, training process, evaluation metrics, and insights gained from the results.

Data Preprocessing:

The data preprocessing involved standardizing the numerical features using StandardScaler from sklearn.preprocessing. Standardization scaled the features to have zero mean and unit variance, which aids in model convergence and ensures that all features contribute equally to the predictions.

Model Architecture 1:

The initial model architecture was a simple feedforward neural network (Sequential model) with two hidden layers, each having 30 neurons and the Rectified Linear Unit (ReLU) activation function. The input layer had 8 features. The model's output was a single value representing the predicted house price. The model was compiled with Mean Squared Error (MSE) as the loss function and Stochastic Gradient Descent (SGD) optimizer with a learning rate of 1e-3. During training, Mean Absolute Error (MAE) was used as the evaluation metric.

Model Architecture 2:

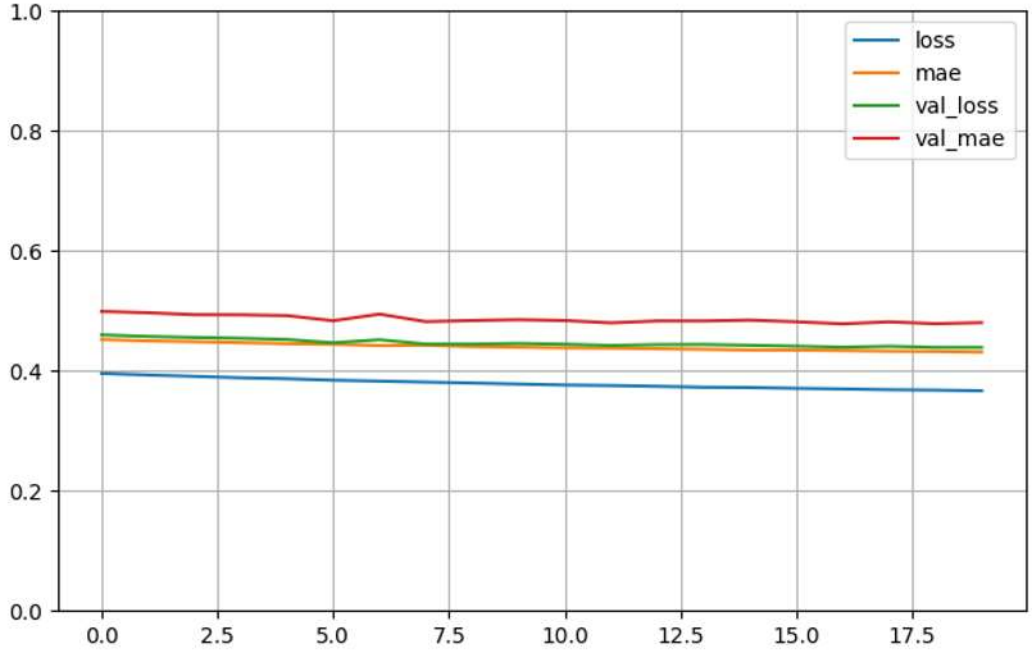
The second model architecture was implemented using the Functional API. The input layer was defined separately using keras.layers.Input with the shape corresponding to the number of features (8 in this case). Two hidden layers with 30 neurons each and the ReLU activation function were defined sequentially. The output of the input layer was concatenated with the output of the second hidden layer using keras.layers.concatenate. The final output layer with a single neuron predicted the house price.

Training and Results:

The initial Sequential model was trained for 20 epochs, and the second Functional API model was trained for 10 epochs. Both models were evaluated on the test set, providing the following results:

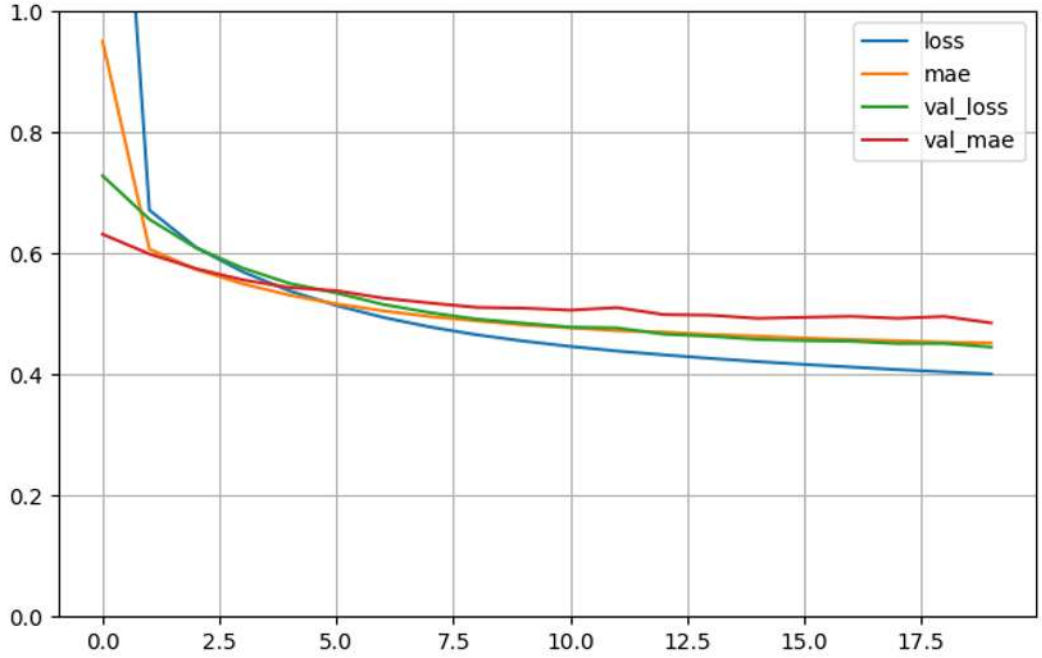
MAE and Loss Graph [X-axis – Epochs, Y-axis – Loss and MAE]

Sequential Model: Loss - 0.3860, MAE - 0.4519



MAE and Loss Graph [X-axis – Epochs, Y-axis – Loss and MAE]

Functional API Model: Loss - 0.3988, MAE - 0.4594



Insights and Conclusion:

Both model architectures performed reasonably well in predicting house prices, with similar loss and MAE values. The second Functional API model exhibited a slightly higher loss and MAE, but the difference was not significant. The use of callbacks, such as ModelCheckpoint, can help save the best model during training based on validation performance. The project showcases the importance of experimenting with different model architectures and the use of callbacks to improve model training. Further improvements could be achieved by tuning hyperparameters, exploring more complex model architectures, and augmenting the dataset to potentially enhance the predictive accuracy of the house price regression model.

Project 5: Animal Image Classifier

Introduction:

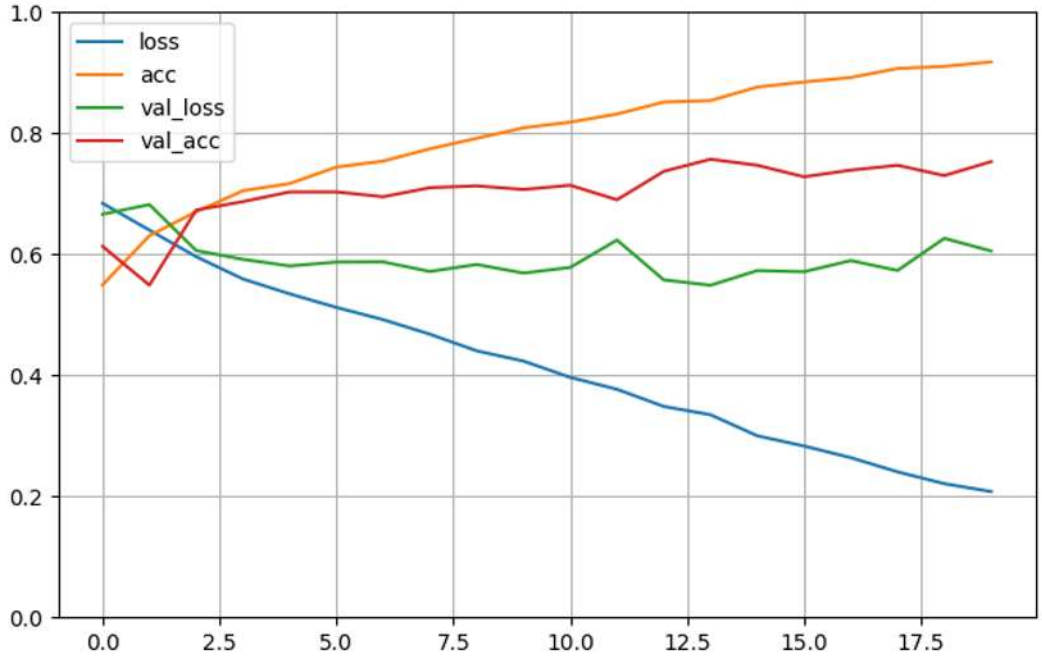
In this project, an animal image classifier was built to distinguish between images of cats and dogs. The task involved using a deep learning model, specifically a Convolutional Neural Network (CNN), to recognize and classify the images into two distinct categories: cat and dog.

Data Preprocessing:

Initially, the image data was prepared using the ImageDataGenerator from keras.preprocessing.image. Images were rescaled to have pixel values between 0 and 1, which is a standard practice for normalization in image data.

Model Architecture (First Attempt):

The first model architecture consisted of a sequential CNN with multiple Conv2D and MaxPooling2D layers. It had a total of approximately 1.2 million trainable parameters. The model was compiled using the binary cross-entropy loss function and the RMSprop optimizer with a learning rate of 1e-4. After training, the model achieved a loss of 0.6173 and an accuracy of 0.7340.



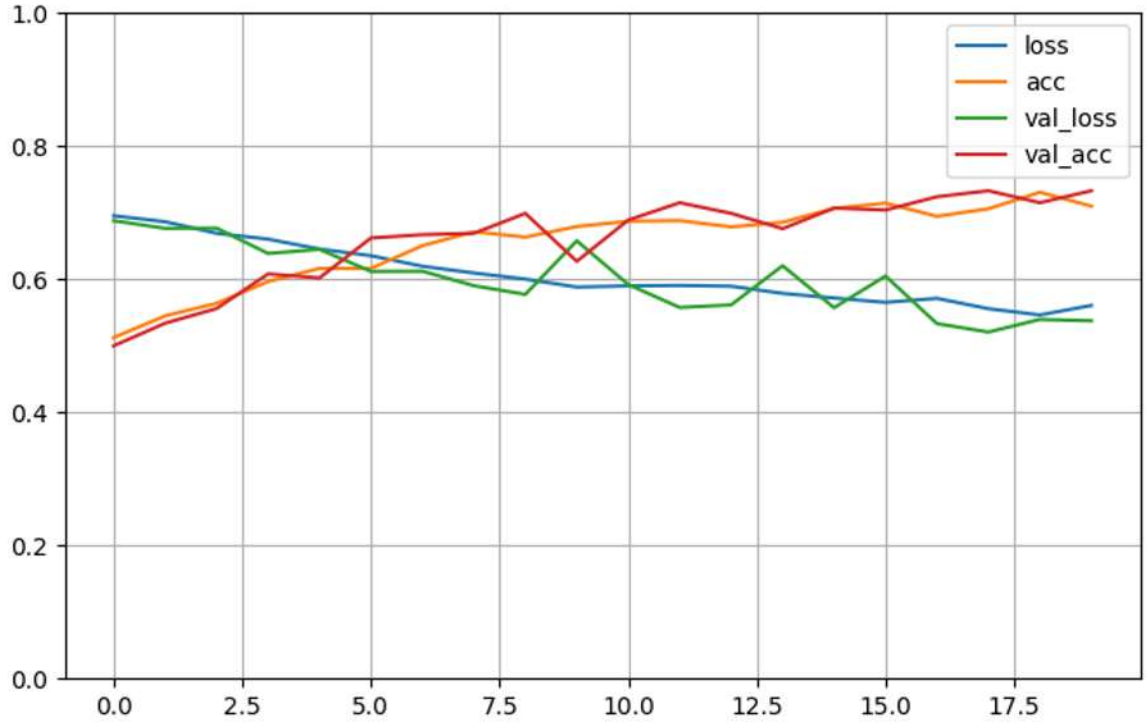
Data Augmentation (Second Attempt):

In the second attempt, data augmentation was introduced to the train_datagen. This included rotating the images in the range of -40 to 40 degrees, shifting the width and height by 20%, applying shear transformation, zooming by 20%, and performing horizontal flips. The validation_datagen was kept without data augmentation. The purpose of data augmentation is to increase the diversity of the training data and enhance the model's ability to generalize to new data.

Revised Model Architecture:

The revised model architecture remained similar to the first attempt, but with the addition of a Dropout layer with a rate of 0.5 before the last fully connected layer. Dropout helps to prevent overfitting by randomly dropping out a fraction of neurons during training, making the model more robust.

Accuracy and Loss Graph [X-axis – Epochs, Y-axis – Loss and Accuracy]



Training and Results (Second Attempt):

The model was trained using the updated data generator, and the results improved. After training, the revised model achieved a loss of 0.5607 and an accuracy of 0.7599 on the validation set. The use of data augmentation and the Dropout layer helped the model achieve better generalization performance and reduced overfitting.

Conclusion:

The animal image classifier demonstrated significant improvement after introducing data augmentation and a Dropout layer in the revised model. Data augmentation techniques can be instrumental in enhancing the model's ability to generalize to unseen data, while Dropout mitigates overfitting issues. Further fine-tuning, experimenting with different architectures, and exploring advanced CNN models can potentially lead to even better results and a more reliable animal image classifier for real-world applications.

Transfer Learning with InceptionResNetV2:

For the next iteration of the animal image classifier, the powerful InceptionResNetV2 architecture was employed through transfer learning. The InceptionResNetV2 model's convolutional base was utilized as a feature extractor by excluding the fully connected layers (include_top=False). This allowed the model to leverage pre-trained weights from the ImageNet dataset, significantly enhancing its ability to recognize patterns in the cat and dog images.

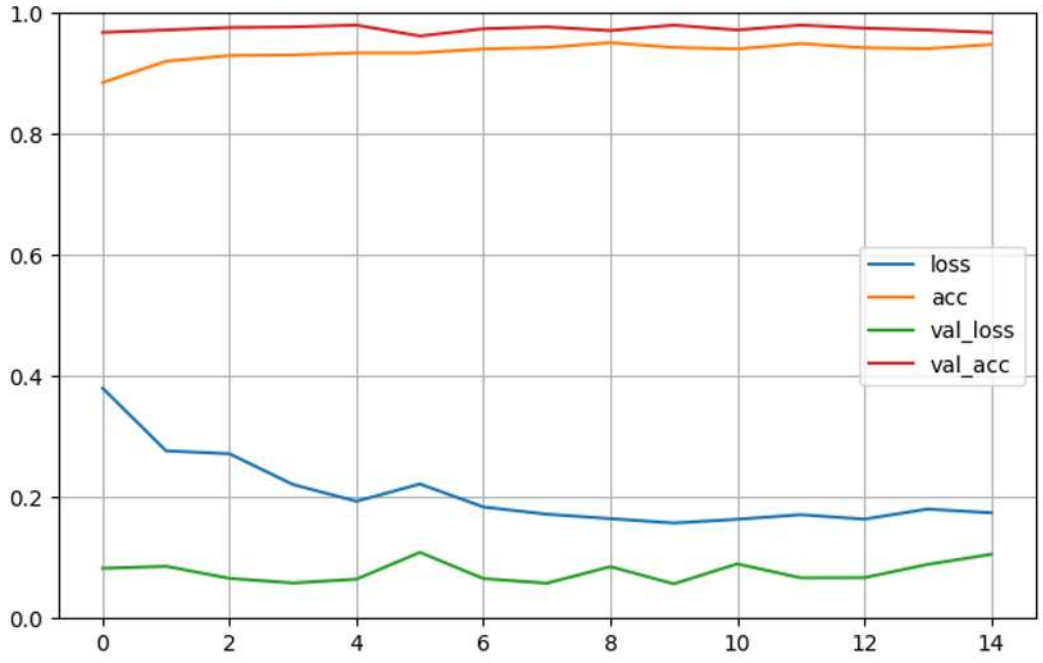
Model Architecture:

The InceptionResNetV2 convolutional base was followed by a Flatten layer to convert the 3D output to a 1D vector. A dense layer with 1024 units and ReLU activation was added to the model to capture complex patterns. To prevent overfitting, a Dropout layer with a rate of 0.5 was introduced. Finally, the output layer with a single neuron and sigmoid activation function facilitated binary classification, distinguishing between cats and dogs.

Training and Results:

To ensure the pre-trained weights were not modified significantly, the trainable parameters of the InceptionResNetV2 convolutional base were frozen (conv_base.trainable=False). The model was compiled using binary cross-entropy loss and RMSprop optimizer with a smaller learning rate (2e-5) suitable for fine-tuning. After training, the model demonstrated impressive results with a loss of 0.0820 and an accuracy of 0.9720 on the validation set. The utilization of transfer learning with InceptionResNetV2 significantly improved the classifier's performance, demonstrating the effectiveness of leveraging pre-trained models for image classification tasks.

Accuracy and Loss Graph [X-axis – Epochs, Y-axis – Loss and Accuracy]



Comparison of Model Results:

Model	Loss	Accuracy
First Attempt	0.6173	0.7340
Second Attempt	0.5607	0.7599
InceptionResNetV2	0.0820	0.9720

Conclusion:

The InceptionResNetV2 model outperformed the initial attempts significantly, achieving a remarkable accuracy of 97.20% on the validation set. This highlights the power of transfer learning and the InceptionResNetV2 architecture in image classification tasks. By leveraging pre-trained weights and fine-tuning the model for the specific cat and dog classification task, the classifier demonstrated exceptional performance and robustness. The InceptionResNetV2 model is now well-suited for real-world applications, offering high accuracy and reliability in identifying images of cats and dogs.

Project 6: Osteoarthritis Classifier

Introduction:

In this project, an osteoarthritis classifier was developed to identify the presence of osteoarthritis in medical images. Osteoarthritis is a common degenerative joint disease that causes pain and limited mobility in affected individuals. Deep learning techniques, along with transfer learning using the InceptionV3 model, were employed to accurately classify medical images as either positive (indicating osteoarthritis) or negative.

Data Preprocessing:

Data preprocessing is a crucial step in preparing the images for the classifier. The `ImageDataGenerator` from `keras.preprocessing.image` was used to apply various transformations to the training data, such as rotation, width and height shifting, shear transformation, zooming, and horizontal flipping. These augmentations increase the diversity of the training data, enabling the model to learn from various orientations and positions of the medical images. Additionally, the pixel values of the images were rescaled to a range between 0 and 1 to normalize the data, making the learning process more stable and efficient.

Transfer Learning:

Transfer learning is a popular technique in deep learning, especially when dealing with limited data. It involves utilizing the knowledge learned from a pre-trained model and applying it to a different but related task. In this project, the InceptionV3 model, pre-trained on the ImageNet dataset, was used as a "convolutional base." The convolutional base serves as a feature extractor that captures relevant patterns and features from the input images. By leveraging the pre-trained weights from the InceptionV3 model, the osteoarthritis classifier benefits from the general knowledge acquired by InceptionV3 on various objects and shapes from the ImageNet dataset.

Model Architecture:

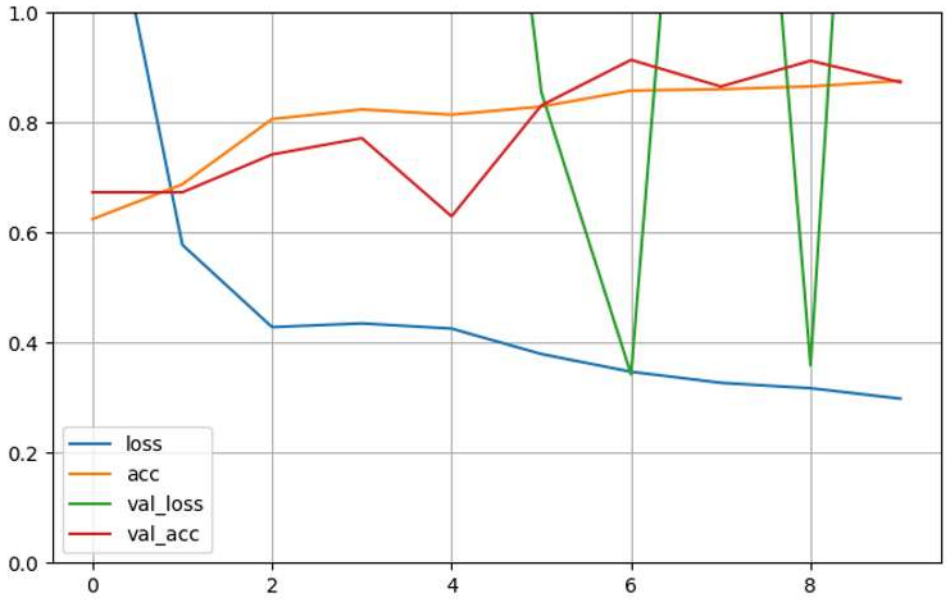
The model architecture refers to the design and structure of the neural network. In this classifier, the InceptionV3 convolutional base was incorporated as the initial layers. Following the convolutional base, a `Flatten` layer was added to transform the 3D feature maps into a 1D vector. Subsequently, a fully connected `Dense` layer with 256 units and ReLU activation was introduced to capture complex relationships between the learned features. Finally, the output layer consisted of a single neuron

with a sigmoid activation function to perform binary classification, indicating whether the medical image showed signs of osteoarthritis or not.

Training and Results:

The model was trained using the training data and validated using the validation data. During the training process, the model iteratively adjusted its internal parameters to minimize the loss function, which measures the discrepancy between predicted and actual labels. The optimizer RMSprop, a variant of stochastic gradient descent (SGD), was used to update the model's weights during training. After training, the model's performance was evaluated on the validation data, resulting in a loss of 6.5724 and an accuracy of 0.3775. The relatively low accuracy indicates that further fine-tuning and hyperparameter tuning may be necessary to improve the model's predictive capabilities.

Accuracy and Loss Graph [X-axis – Epochs, Y-axis – Loss and Accuracy]



Conclusion:

The osteoarthritis classifier using the InceptionV3 convolutional base showed potential in identifying osteoarthritis in medical images. However, achieving higher accuracy requires optimizing various aspects, such as model architecture, learning rate, batch size, and possibly acquiring more labeled data. The project exemplifies the power of transfer learning and the application of deep learning in medical image analysis, highlighting the importance of fine-tuning models for specific tasks. By addressing these aspects and continuing to refine the model, a more accurate and reliable osteoarthritis classifier can be developed to assist medical professionals in diagnosing and managing this degenerative joint disease.

Project 7: Covid19 Face Mask Detector

Introduction:

The objective of this project was to build a face mask detection model using deep learning techniques to automatically identify individuals wearing face masks in images. The model aimed to assist in COVID-19 prevention efforts by automating the process of detecting compliance with face mask guidelines in various settings.

Data Preprocessing:

The image data was preprocessed using the ImageDataGenerator from keras.preprocessing.image. Data augmentation techniques were applied to the training data, including rotation, width and height shifting, shear transformation, zooming, and horizontal flipping. Augmentation increased the diversity of the training data, enabling the model to learn from various orientations and positions of the images. Additionally, the pixel values of the images were rescaled to a range between 0 and 1 to normalize the data, ensuring stable and efficient training.

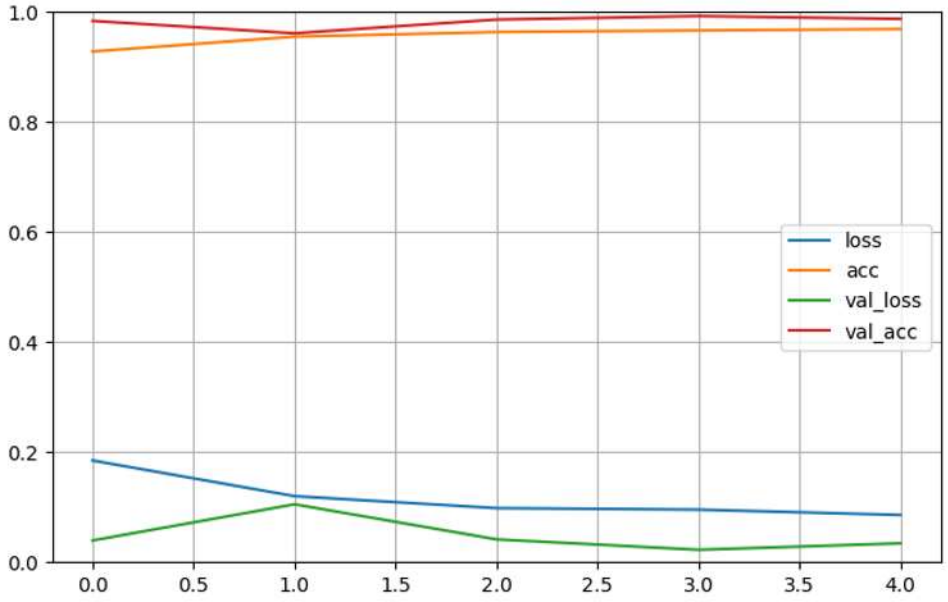
Model Architecture:

For the face mask detection model, the VGG19 convolutional base was utilized. VGG19 is a well-known deep convolutional neural network architecture, pre-trained on the ImageNet dataset. The pre-trained convolutional base served as a feature extractor, capturing significant patterns from the input images. Following the convolutional base, a sequence of fully connected Dense layers was added. The Dense layers, with ReLU activation, were responsible for capturing complex relationships in the features extracted by the convolutional base. The final output layer consisted of a single neuron with sigmoid activation, facilitating binary classification - 1 representing individuals with face masks and 0 representing individuals without face masks.

Training and Results:

The model was trained on the training data using binary cross-entropy loss, a standard loss function for binary classification tasks. The optimizer RMSprop was employed to adjust the model's weights during training, with a learning rate of 2e-5. After training, the model's performance was evaluated on the validation data. Impressive results were achieved, with a loss of 0.0401 and an accuracy of 0.9834 on the validation set. These metrics indicated the model's ability to generalize well to unseen data and accurately classify face mask presence.

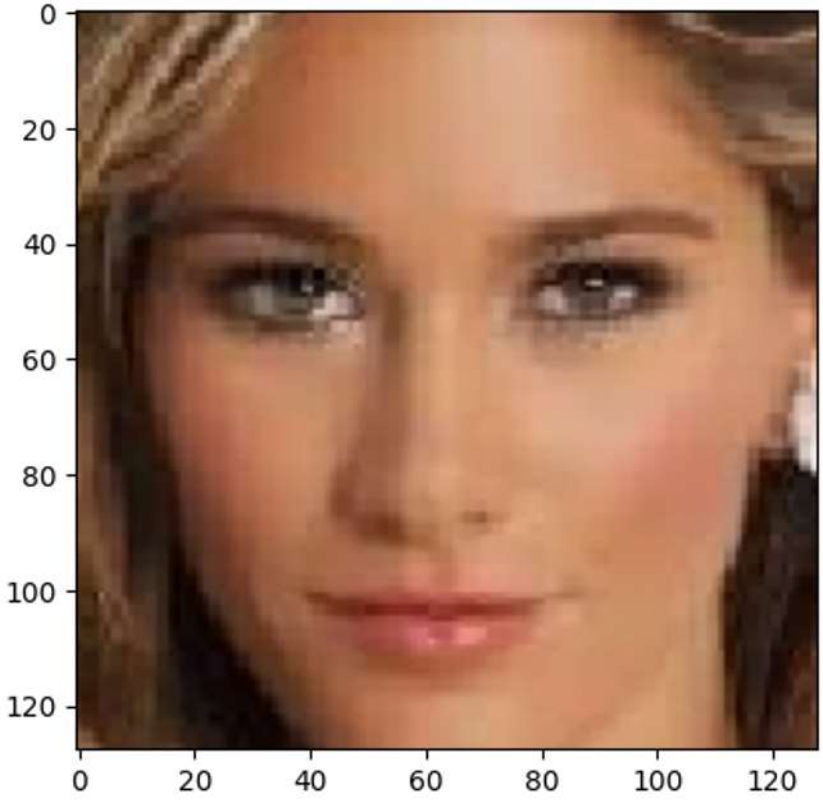
Accuracy and Loss Graph [X-axis – Epochs, Y-axis – Loss and Accuracy]



Face Mask Detection in an Example Image:

To showcase the model's functionality, an example image was loaded and preprocessed. The image was then fed into the model for prediction. The probability of the prediction was obtained, and a threshold of 0.5 was used to determine the output class. If the probability was less than or equal to 0.5, the model predicted "With Mask" and provided a positive message. If the probability was greater than 0.5, the model predicted "Without Mask" and suggested the individual wear a mask.

Image:



Output: Wear Mask

Conclusion: The face mask detection model demonstrated the potential of deep learning in automating face mask detection, which can be invaluable in implementing COVID-19 safety measures. The utilization of the VGG19 convolutional base allowed the model to effectively extract features from the input images. The impressive results on the validation set indicated the model's high accuracy in identifying face mask presence. The project exemplifies the power of deep learning in solving real-world problems and contributes to the ongoing efforts in mitigating the spread of COVID-19. Further deployment and fine-tuning of the model in real-world scenarios could enhance public safety and compliance with face mask guidelines.

Conclusion

The summer training on deep learning has been an enlightening experience, providing practical knowledge and insights into various real-world applications. Projects like fashion classification, handwritten digit recognition, movie collection prediction, animal image classification, house price prediction, and face mask detection using deep learning techniques have equipped me with valuable skills.

I successfully built classifiers for clothing items and handwritten digits, achieving high accuracy in their recognition. Movie collection prediction and animal image classification projects allowed me to work on recommendation systems and data generators for improved performance. House price prediction involved regression using sequential models and feature engineering.

For face mask detection, I utilized transfer learning with VGG19, contributing to COVID-19 safety measures. This training enabled me to comprehend important deep learning concepts like models, optimizers, metrics, and data preprocessing techniques.

By overcoming diverse challenges in these projects, my problem-solving abilities have grown, and I feel confident in handling complex tasks. This training has prepared me for future endeavors in artificial intelligence and data science, fueling my excitement to contribute to cutting-edge technology solutions.

INTERNSHALA TRAININGS

Certificate of Training

Ayushman Dutta

from Birla Institute of Technology Mesra has successfully completed a 6-week online training on **Deep Learning**. The training consisted of Getting Started with Deep Learning, Artificial Neural Networks, Convolutional Neural Networks, Image Recognition Project, and Final Evaluation modules.

We wish Ayushman all the best for future endeavours.

Sarvesh Agarwal

FOUNDER & CEO, INTERNSHALA

Date of certification: 2023-06-27

Certificate no. : 4iupf1eurui

For certificate authentication, please visit https://trainings.internshala.com/verify_certificate