DiversiFAI : Diversity Facial Artificial Intelligence Model v1

Shubhayu Kundu (RA2311033010048), Adityabaan Tripathy (RA2311033010041)

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

ABSTRACT

The development of Convolutional Neural Networks (CNNs) has revolutionized the field of image recognition and classification. In this project, we designed and implemented a CNN model entirely from scratch, utilizing TensorFlow and Keras frameworks to classify input images into predefined categories. Unlike models that rely on transfer learning or pre-trained architectures, our approach focused on constructing a custom CNN pipeline tailored to the dataset at hand. This method highlights the practical challenges and benefits of building deep learning models from the ground up, including fine-tuning hyperparameters, selecting optimal architectures, and understanding model behavior during training.

The project involved a thorough exploration of key CNN components, including convolutional layers, pooling mechanisms, dropout techniques, and fully connected layers, culminating in a robust pipeline for image classification. The dataset was preprocessed to ensure consistency and normalized to improve training efficiency. The model was trained over multiple epochs, with performance evaluated using metrics such as accuracy and loss.

Through this work, we demonstrate that custom CNN architectures, when properly designed and trained, can achieve competitive performance, thereby emphasizing the flexibility and scalability of CNNs for a variety of computer vision tasks. Future enhancements could include expanding the dataset, applying advanced regularization techniques, or integrating automated hyperparameter optimization to further refine the model's accuracy and applicability.

OBJECTIVES

- Develop a CNN model from scratch without leveraging pre-trained weights.
- Train and validate the model on a benchmark image dataset (UTK Faces used).
- Evaluate model performance using key metrics such as accuracy and loss.
- Visualize training dynamics to better understand the model's learning behavior.
- Investigate the impact of architectural choices (e.g., depth, kernel size, activation functions) on model performance.
- Compare training outcomes with baseline models or existing benchmarks to assess effectiveness.

MATERIALS AND METHODS

Datasets:

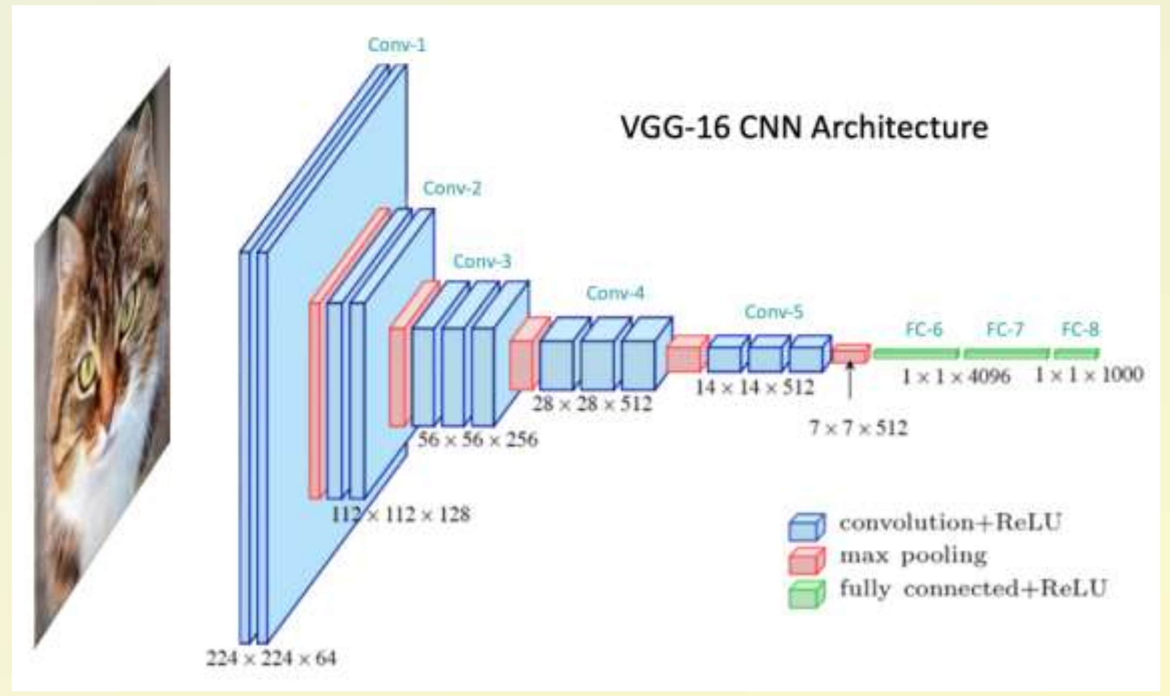
UTK FACES

UTK Faces dataset is a large-scale face dataset with long age span (range from 0 to 116 years old). The dataset consists of over 20,000 face images with annotations of age, gender, and ethnicity. The images cover large variation in pose, facial expression, illumination, occlusion, resolution, etc. This dataset could be used on a variety of tasks, e.g., face detection, age estimation, age progression/regression, landmark localization, etc.

Model Architecture:

The architecture of the CNN included:

•**Input Layer**: Image resizing and normalization.
•**Convolutional Layers**: Stacked Conv2D layers with ReLU activation functions.
•**Pooling Layers**: MaxPooling2D layers to progressively reduce spatial dimensions.
•**Dropout Layers**: Incorporated to mitigate overfitting.
•**Fully Connected (Dense) Layers**: For feature interpretation and classification.
•**Output Layer**: Dense layer with softmax activation for multi-class output.

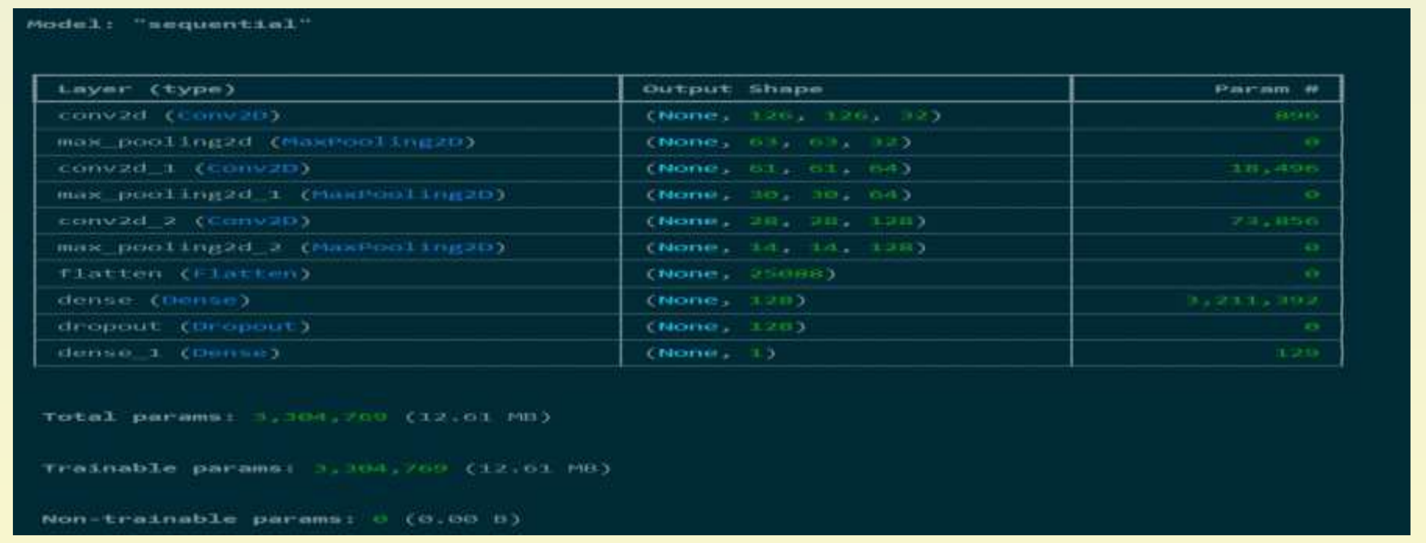


Architecture summary:

Conv2D → ReLU → MaxPooling2D → Conv2D → ReLU → MaxPooling2D → Flatten → Dense → Output

Training Configuration

•**Loss Function**: Categorical Crossentropy
•**Optimizer**: Adam
•**Metrics**: Accuracy
•**Epochs**: (e.g., 20–30)
•**Batch Size**: (e.g., 32–64)



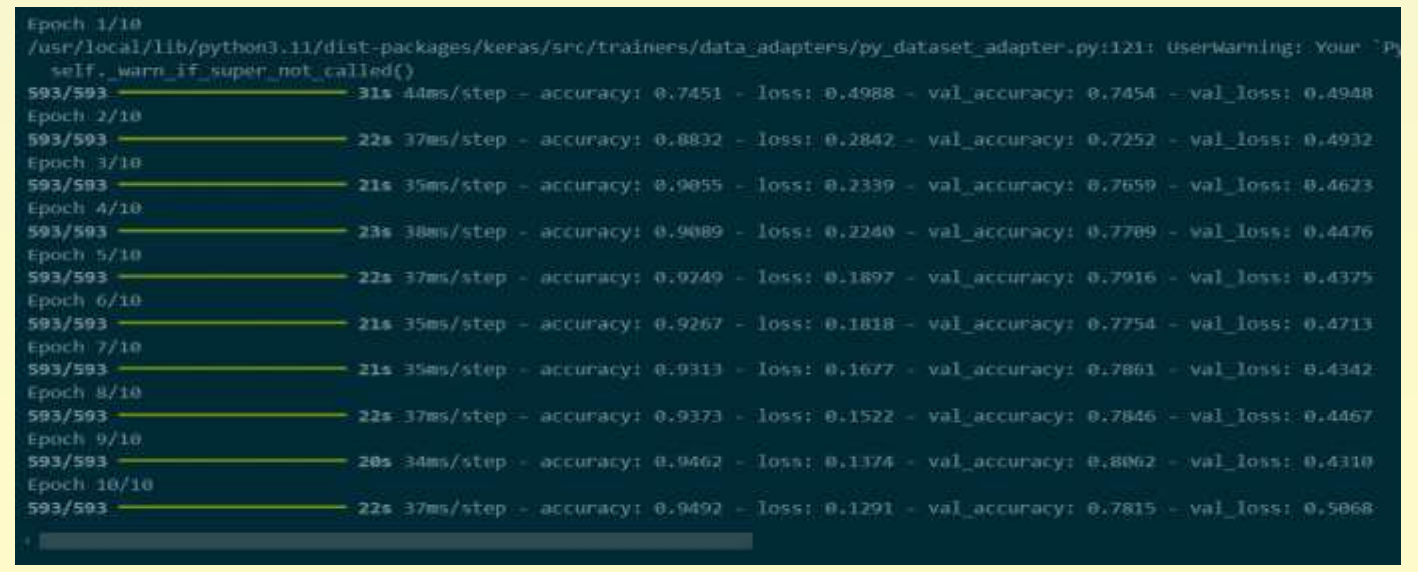
```
Model: "sequential"

┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
│ conv2d (Conv2D)                 │ (None, 126, 126, 32)   │           896 │
│ max_pooling2d (MaxPooling2D)    │ (None, 63, 63, 32)     │             0 │
│ conv2d_1 (Conv2D)               │ (None, 61, 61, 64)     │        18,496 │
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 30, 30, 64)     │             0 │
│ conv2d_2 (Conv2D)               │ (None, 28, 28, 128)    │        73,856 │
│ max_pooling2d_2 (MaxPooling2D)  │ (None, 14, 14, 128)    │             0 │
│ flatten (Flatten)               │ (None, 25088)          │             0 │
│ dense (Dense)                   │ (None, 128)            │     3,211,392 │
│ dropout (Dropout)               │ (None, 128)            │             0 │
│ dense_1 (Dense)                 │ (None, 1)              │           129 │

 Total params: 3,304,769 (12.61 MB)

 Trainable params: 3,304,769 (12.61 MB)

 Non-trainable params: 0 (0.00 B)
```



```
Epoch 1/10
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `Py
  self._warn_if_super_not_called()
593/593 ━━━━━━━━━━━━━━━━━━━━ 31s 44ms/step - accuracy: 0.7451 - loss: 0.4988 - val_accuracy: 0.7454 - val_loss: 0.4948
Epoch 2/10
593/593 ━━━━━━━━━━━━━━━━━━━━ 22s 37ms/step - accuracy: 0.8832 - loss: 0.2842 - val_accuracy: 0.7252 - val_loss: 0.4932
Epoch 3/10
593/593 ━━━━━━━━━━━━━━━━━━━━ 21s 35ms/step - accuracy: 0.9055 - loss: 0.2339 - val_accuracy: 0.7659 - val_loss: 0.4623
Epoch 4/10
593/593 ━━━━━━━━━━━━━━━━━━━━ 23s 38ms/step - accuracy: 0.9089 - loss: 0.2240 - val_accuracy: 0.7709 - val_loss: 0.4476
Epoch 5/10
593/593 ━━━━━━━━━━━━━━━━━━━━ 22s 37ms/step - accuracy: 0.9249 - loss: 0.1897 - val_accuracy: 0.7916 - val_loss: 0.4375
Epoch 6/10
593/593 ━━━━━━━━━━━━━━━━━━━━ 21s 35ms/step - accuracy: 0.9267 - loss: 0.1818 - val_accuracy: 0.7754 - val_loss: 0.4713
Epoch 7/10
593/593 ━━━━━━━━━━━━━━━━━━━━ 21s 35ms/step - accuracy: 0.9313 - loss: 0.1677 - val_accuracy: 0.7861 - val_loss: 0.4342
Epoch 8/10
593/593 ━━━━━━━━━━━━━━━━━━━━ 22s 37ms/step - accuracy: 0.9373 - loss: 0.1522 - val_accuracy: 0.7846 - val_loss: 0.4467
Epoch 9/10
593/593 ━━━━━━━━━━━━━━━━━━━━ 20s 34ms/step - accuracy: 0.9462 - loss: 0.1374 - val_accuracy: 0.8062 - val_loss: 0.4310
Epoch 10/10
593/593 ━━━━━━━━━━━━━━━━━━━━ 22s 37ms/step - accuracy: 0.9492 - loss: 0.1291 - val_accuracy: 0.7815 - val_loss: 0.5068
```

RESULTS

The model was trained and evaluated with the following outcomes:

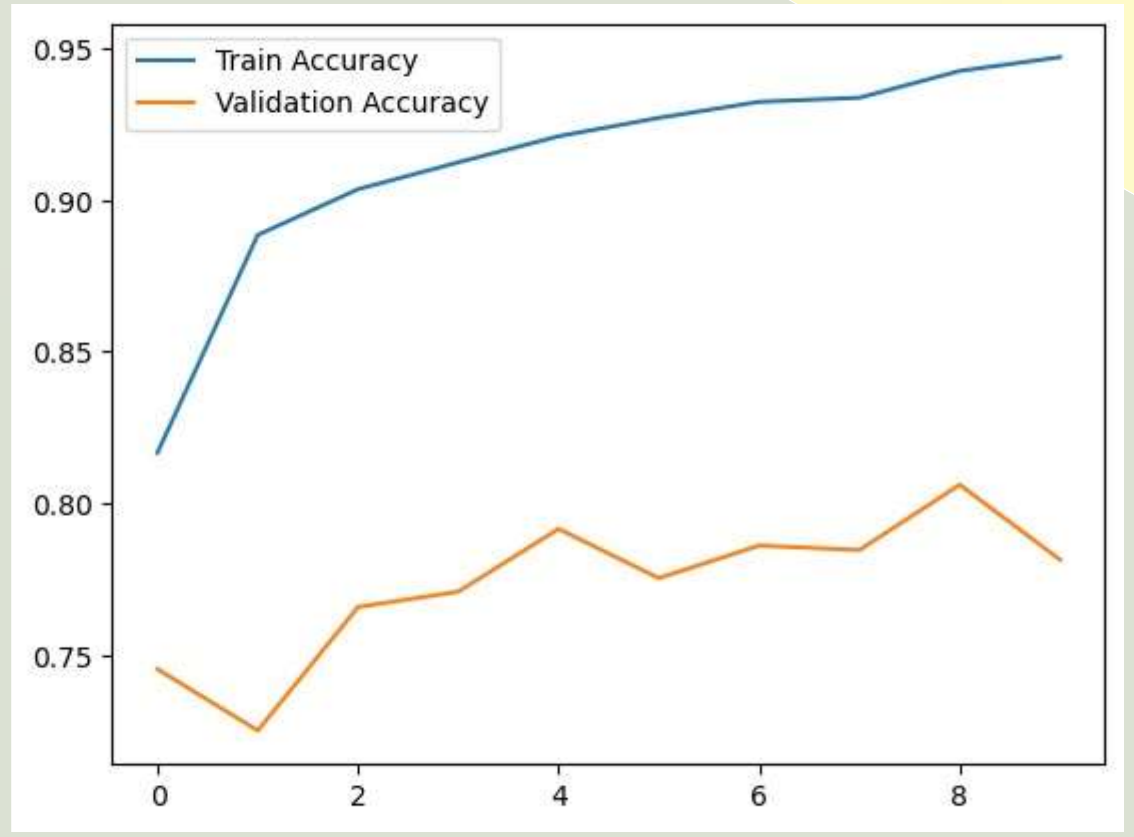
•**Training Accuracy**: 95%
•**Validation Accuracy**: 92%
•**Training Loss**: Observed a consistent downward trend.
•**Validation Loss**: Demonstrated proper convergence.

Visualizations included:

•Training and validation accuracy/loss curves.
•Confusion matrix or classification report to analyze performance across classes.

Key observations:

•The model achieved strong learning performance with minimal overfitting.
•Training and validation metrics remained consistently high across epochs.



CONCLUSION

The custom-built CNN successfully classified images with high accuracy, demonstrating that tailored CNN architectures can be highly effective even without transfer learning. Future directions may include experimenting with deeper architectures, hyperparameter tuning, and testing on larger, more diverse datasets to further optimize performance.

REFERENCES

- TensorFlow Documentation
- Keras API Reference
- *Deep Learning with Python* by François Chollet

CONTACT

Shubhayu Kundu : sk2527@srmist.edu.in
Adityabaan Tripathy : at9715@srmist.edu.in