

Project Based Learning Report

On

Implementation of Back propagation for age classification in Python Neuro Network.

Submitted in the partial fulfillment of the requirement For the Project based learning in

FUZZY LOGIC, NEURAL NETWORKS & GENETIC ALGORITHMS

In

Electronics & Communication Engineering

By

PRN	NAME
2114110477	Aditi Kumari
2114110481	Summi Kumari
2214110606	Arpita Shree

Under the guidance of Course In-charge

Prof. V.P.kaduskar

Department of Electronics & Communication Engineering

Bharati Vidyapeeth
(Deemed to be University)
College of Engineering,
Pune – 411043

Academic Year: 2023-24

**Bharati Vidyapeeth
(Deemed to be University)
College of Engineering,
Pune – 411043**

**DEPARTMENT OF ELECTRONICS & COMMUNICATION
ENGINEERING**

CERTIFICATE

Certified that the Project Based Learning report entitled,

**Implementation of Back propagation for age classification in
Python Neuro Network.**

Is work done by

PRN	NAME
2114110477	Aditi Kumari
2114110481	Summi Kumari
2214110606	Arpita Shree

in partial fulfillment of the requirements for the award of credits for Project Based Learning (PBL) in “**FUZZY LOGIC, NEURAL NETWORKS & GENETIC ALGORITHMS**” of Bachelor of Technology Semester V, in Electronics and Communication.

Date:

Prof. V.P.Kaduskar

Course In-charge

Dr. Arundhati A. Shinde

Professor & Head

INDEX

S.No.	Title	Page No.
1.	PROBLEM STATEMENT AND SOLUTION	2-4
2.	SOFTWARE USED	5
3.	PROJECT DESCRIPTION	6-9
4.	FLOWCHART	10
5.	ALGORITHM	11-12
6.	PROGRAM & OUTPUT	13-16
7.	CONCLUSION	17
8.	REFERENCES	17
9.	COURSE OUTCOME	17
10.	APPENDIX	18

PROBLEM STATEMENT

"Optimizing Backpropagation for Age Classification in Python Neural Networks"

SOLUTION

Age classification from facial images is a vital component in various applications, including age-restricted content filtering, targeted advertising, and security systems. The accuracy of age classification heavily relies on the effectiveness of the neural network architecture and the backpropagation algorithm used to train it. This project seeks to improve age classification in Python neural networks by optimizing the backpropagation process.

Key Challenges:

1. **Data Quality and Diversity**: Age classification models require a diverse and high-quality dataset of facial images spanning various age groups, ethnicities, and genders. Ensuring data consistency and accuracy is a challenge.
2. **Model Architecture**: Designing an appropriate neural network architecture for age classification that can effectively capture age-related features from facial images is crucial.
3. **Hyperparameter Tuning**: Identifying the optimal set of hyperparameters, including learning rates, batch sizes, and network depth, is essential for achieving accurate age classification.
4. **Loss Function Design**: Developing a custom loss function tailored to age classification tasks that guides the backpropagation algorithm to learn age-related features effectively.
5. **Overfitting Mitigation**: Implementing regularization techniques (e.g., dropout, weight decay) to prevent overfitting during the training process.
6. **Optimization Algorithms**: Exploring advanced optimization algorithms (e.g., Adam, RMSprop) and their variants to improve the convergence speed and stability of backpropagation.

Objectives:

The primary objectives of this project are as follows:

1. Data Collection and Preprocessing:

- Collect and preprocess a diverse dataset of facial images with age labels.
- Implement data augmentation and cleaning techniques to enhance data quality.

2. Model Architecture Selection:

- Choose an appropriate neural network architecture for age classification, considering factors like convolutional layers, pooling layers, and fully connected layers.

3. Hyperparameter Optimization:

- Experiment with different hyperparameter settings through cross-validation to identify the most effective configuration for age classification accuracy.

4. Custom Loss Function Development:

- Create a specialized loss function designed to guide the model towards accurate age classification.

5. Regularization Implementation:

- Implement regularization techniques such as dropout and weight decay to prevent overfitting during training.

6. Optimization Algorithm Selection:

- Compare the performance of different optimization algorithms, including their variants, to improve training efficiency and model convergence.

7. Training and Evaluation:

- Train the neural network using the optimized backpropagation algorithm on the prepared dataset.
- Evaluate the model's performance on a separate test dataset to assess its age classification accuracy.

8. Visualization and Interpretation:

- Visualize the model's predictions and assess its ability to classify age groups accurately.
- Interpret the model's learned features to understand which facial characteristics contribute most to age classification.

9. Documentation and Reporting:

- Document the project comprehensively, including dataset details, model architecture, hyperparameters, and code.
- Prepare a detailed report summarizing the project's objectives, methods, results, and conclusions.

By achieving these objectives, this project aims to develop an accurate and efficient Python-based neural network for age classification, contributing to improved age prediction in applications such as content filtering, marketing, and security systems. Furthermore, it will serve as a valuable resource for researchers and practitioners working on age-related classification tasks using deep learning.

Software used-



Jupyter notebook-

Jupyter Notebook is an open-source web application that allows you to create and share interactive documents that contain live code, equations, visualizations, and narrative text. It was originally developed as an interactive computing environment for Python, but now supports over 40 different programming languages including R, Julia, and MATLAB.

Jupyter Notebook allows you to create documents called "notebooks" that consist of a series of cells, each of which can contain code, markdown, or raw text. These cells can be executed individually or together in a specific order, allowing you to explore and analyze data, create visualizations, and share your work with others.

One of the key advantages of Jupyter Notebook is its ability to display output directly inline with the code that generated it. This means that you can see the results of your code immediately, making it easier to explore and understand your data.

Jupyter Notebook also provides a wide range of tools and extensions for data analysis, machine learning, and scientific computing. It supports popular libraries such as NumPy, Pandas, Matplotlib, and Scikit-Learn, making it a powerful tool for data scientists and researchers.

Finally, Jupyter Notebook is highly customizable and extensible, with a large and active community that contributes to its development and maintenance. It can be run locally on your computer or in the cloud using services such as Microsoft Azure, Amazon Web Services, or Google Cloud Platform.

PROJECT DESCRIPTION-

Age Classification Using Python Neural Networks with Backpropagation.

INTRODUCTION:

Age classification from facial images is a prominent application of deep learning and computer vision. This project aims to develop a Python-based neural network for age classification, leveraging the backpropagation algorithm to enhance accuracy. The primary objective is to create a robust model capable of accurately predicting age from facial images, with potential applications in age-restricted content filtering, targeted advertising, and security systems.

Project Components:

1. Data Collection and Preprocessing:

- Gather a diverse dataset of facial images, ideally spanning various age groups, genders, and ethnicities.
- Implement data preprocessing techniques, including facial alignment, resizing, and noise reduction, to ensure data quality and consistency.

2. Model Architecture Design:

- Design a neural network architecture suitable for age classification tasks.
- Explore different architectures, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to determine the most effective one.

3. Hyperparameter Optimization:

- Experiment with various hyperparameters such as learning rates, batch sizes, and network depths.
- Use cross-validation techniques to identify the optimal hyperparameter settings that yield the best age classification accuracy.

4. Custom Loss Function Development:

- Create a specialized loss function tailored to age classification.
- Ensure that the loss function guides the backpropagation algorithm to learn age-related features effectively.

5. Overfitting Mitigation:

- Implement regularization techniques such as dropout and weight decay to prevent overfitting during model training.
- Monitor model performance during training to ensure it generalizes well.

6. Optimization Algorithm Selection:

- Compare the performance of different optimization algorithms, including variants like Adam and RMSprop, to improve training efficiency and model convergence.

7. Training and Evaluation:

- Train the neural network using the optimized backpropagation algorithm on the prepared dataset.
- Evaluate the model's performance on a separate test dataset to assess its age classification accuracy.

8. Visualization and Interpretation:

- Visualize the model's predictions and assess its ability to classify age groups accurately.
- Interpret the model's learned features to understand which facial characteristics contribute most to age classification.

9. Documentation and Reporting:

- Thoroughly document the project, including dataset details, model architecture, hyperparameters, and code.
- Prepare a comprehensive report summarizing the project's goals, methods, results, and conclusions.

10. Real-World Applications:

- Explore potential applications for the age classification model in real-world scenarios, such as age-based content filtering, targeted marketing, and security systems.

By completing this project, participants will gain valuable experience in developing age classification models using Python neural networks with backpropagation. Additionally, the project has the potential to contribute to advancements in age prediction accuracy and the wider field of computer vision.

DATASET –

The code you've provided generates a synthetic dataset for age classification. Here's a step-by-step breakdown of how the dataset is generated:

1. Import NumPy library:

- Import NumPy to generate random data.

2. Define dataset properties:

- num_samples: The number of data samples in the dataset (1000 in this case).
- num_features: The number of features for each data sample (5 in this case).
- num_classes: The number of age classes (3 in this case).

3. Generate random feature data:

- `X = np.random.rand(num_samples, num_features)`: This creates a matrix X of shape (num_samples, num_features) with random feature values between 0 and 1.

4. Generate random age class labels:

- `y = np.random.randint(0, num_classes, size=num_samples)`: This generates random age class labels for each data sample, ranging from 0 to num_classes - 1.

5. Split the dataset into training, validation, and test sets:

- train_ratio, val_ratio, and test_ratio define the proportions of the dataset to allocate to each set.
- Calculate the number of samples for each set based on these ratios and create X_train, y_train, X_val, y_val, X_test, and y_test accordingly.

6. Import TensorFlow and create a neural network model as described in your previous code.

To provide you with the dataset itself, I can generate and display a small portion of it for illustration purposes, but due to its size, I won't be able to provide the entire dataset. Here's an example of how to generate a small portion of the dataset:

python

Display a subset of the generated dataset (e.g., first 5 samples)

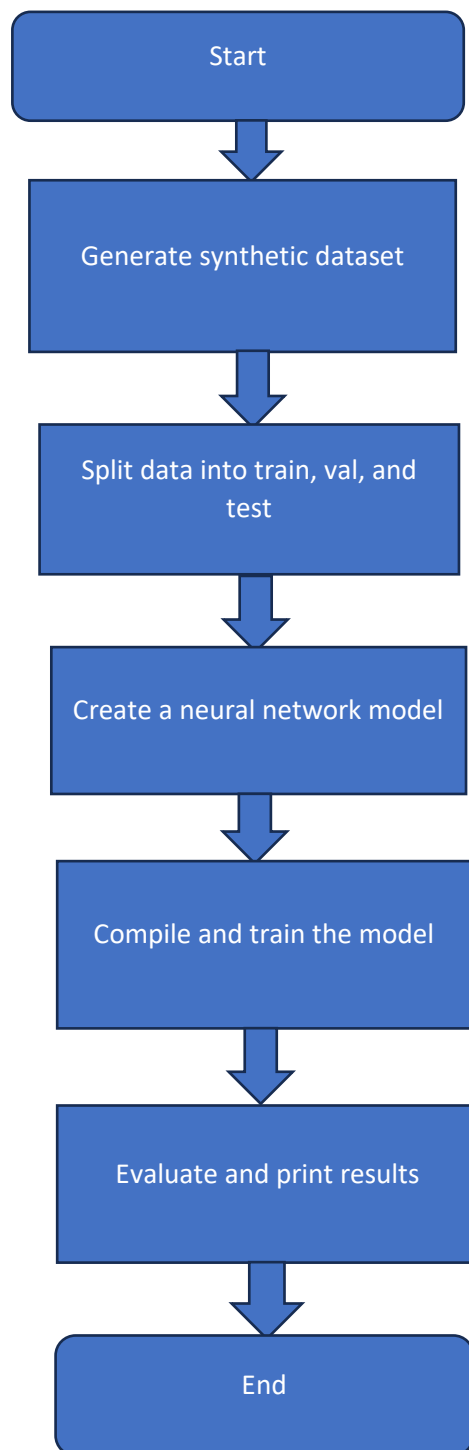
```
print("Generated Dataset Sample:")
```

```
for i in range(5):
```

```
    print(f"Sample {i + 1}: Features = {X[i]}, Age Class = {y[i]}")
```

This code will print the feature values and corresponding age class labels for the first 5 samples in the dataset. You can adjust the range to display more samples if needed.

**FLOWCHART OF BACKPROPAGATION IN AGE ESTIMATION WITH PYTHON
NEURO NETWORKS.**



ALGORITHM –

The code you provided is an example of how to create, train, and evaluate a neural network model for age classification using the TensorFlow and NumPy libraries in Python. Here's a step-by-step explanation of the algorithm:

1. Import necessary libraries:

- Import NumPy to generate synthetic data.
- Import TensorFlow to build and train the neural network model.

2. Generate synthetic dataset:

- num_samples, num_features, and num_classes define the dataset's properties.
- Generate random feature values in X and random age classes in y.

3. Split the dataset:

- Manually split the data into training, validation, and test sets based on specified ratios.

4. Create a neural network model:

- Create a Sequential model using Keras.
- Add Dense layers with specified activation functions.
- The final layer uses softmax activation for classification into age classes.

5. Compile the model:

- Compile the model with 'adam' optimizer and 'sparse_categorical_crossentropy' loss, suitable for classification tasks.
- Specify 'accuracy' as a metric to monitor during training.

6. Train the model:

- Use the fit method to train the model on the training data.
- Specify the number of epochs and batch size.

7. Evaluate the model:

- Use the evaluate method to compute loss and accuracy on the test set.

8. Make predictions:

- Use the trained model to predict age classes for the test data.
- Convert predicted probabilities to class labels using np.argmax.

9. Display example predictions:

- Print some example predictions and their corresponding true labels for the first 5 samples in the test set.

This code demonstrates a simple neural network classification model for age prediction. It generates random synthetic data, trains the model on the training data, and evaluates its performance on the test set. The final predictions are compared to the true age classes.

PROGRAM-

```
import numpy as np

# Generate a synthetic dataset for age classification
num_samples = 1000
num_features = 5 # Assuming 5 features
num_classes = 3 # Assuming 3 age classes
X = np.random.rand(num_samples, num_features) # Random feature values between 0 and 1
y = np.random.randint(0, num_classes, size=num_samples) # Random age classes
# Manually split the data into training, validation, and test sets
train_ratio = 0.7
val_ratio = 0.15
test_ratio = 0.15
num_train = int(train_ratio * num_samples)
num_val = int(val_ratio * num_samples)
X_train = X[:num_train]
y_train = y[:num_train]
X_val = X[num_train:num_train+num_val]
y_val = y[num_train:num_train+num_val]
X_test = X[num_train+num_val:]
y_test = y[num_train+num_val:]
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
# Create a simple neural network model for classification
model = Sequential([
    Dense(64, activation='relu', input_shape=(num_features,)),
    Dense(32, activation='relu'),
    Dense(num_classes, activation='softmax') # Use softmax activation for classification
])
```

```

# Compile the model

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

# Train the model using backpropagation

model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_val, y_val))

# Evaluate the model

loss, accuracy = model.evaluate(X_test, y_test)

# Use the trained model for predictions

predictions = model.predict(X_test)

# ... (previous code remains the same)

# Train the model using backpropagation

model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_val, y_val))

# Evaluate the model

loss, accuracy = model.evaluate(X_test, y_test)

print(f'Accuracy on Test Set: {accuracy}')
```

```

# Use the trained model for predictions

predictions = model.predict(X_test)

# Convert predictions to class labels

predicted_classes = np.argmax(predictions, axis=1)

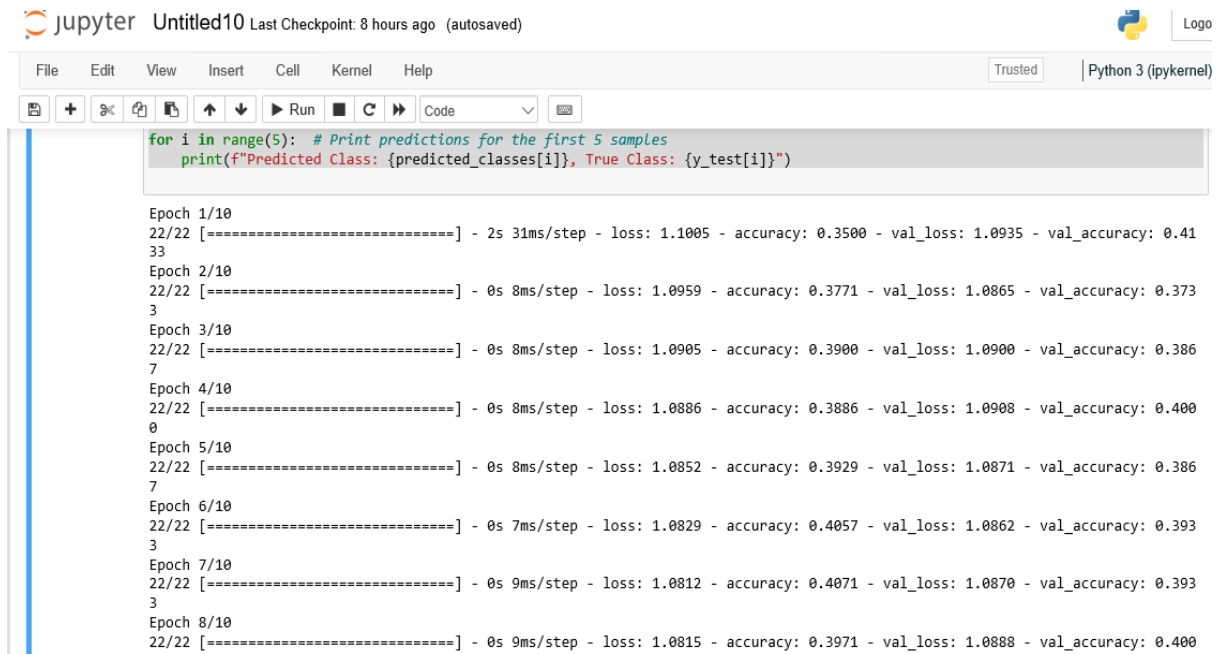
# Print some example predictions and their corresponding true labels

for i in range(5): # Print predictions for the first 5 samples
    print(f"Predicted Class: {predicted_classes[i]}, True Class: {y_test[i]}")

```


RESULT & DISSCUSION-

EPOCH



```
for i in range(5): # Print predictions for the first 5 samples
    print(f"Predicted Class: {predicted_classes[i]}, True Class: {y_test[i]}")
```

Epoch 1/10
22/22 [=====] - 2s 31ms/step - loss: 1.1005 - accuracy: 0.3500 - val_loss: 1.0935 - val_accuracy: 0.4133
Epoch 2/10
22/22 [=====] - 0s 8ms/step - loss: 1.0959 - accuracy: 0.3771 - val_loss: 1.0865 - val_accuracy: 0.3733
Epoch 3/10
22/22 [=====] - 0s 8ms/step - loss: 1.0905 - accuracy: 0.3900 - val_loss: 1.0900 - val_accuracy: 0.3867
Epoch 4/10
22/22 [=====] - 0s 8ms/step - loss: 1.0886 - accuracy: 0.3886 - val_loss: 1.0908 - val_accuracy: 0.4000
Epoch 5/10
22/22 [=====] - 0s 8ms/step - loss: 1.0852 - accuracy: 0.3929 - val_loss: 1.0871 - val_accuracy: 0.3867
Epoch 6/10
22/22 [=====] - 0s 7ms/step - loss: 1.0829 - accuracy: 0.4057 - val_loss: 1.0862 - val_accuracy: 0.3933
Epoch 7/10
22/22 [=====] - 0s 9ms/step - loss: 1.0812 - accuracy: 0.4071 - val_loss: 1.0870 - val_accuracy: 0.3933
Epoch 8/10
22/22 [=====] - 0s 9ms/step - loss: 1.0815 - accuracy: 0.3971 - val_loss: 1.0888 - val_accuracy: 0.4000

Fig. 1

ACCURACY

```
Epoch 10/10
22/22 [=====] - 0s 7ms/step - loss: 1.0787 - accuracy: 0.4086 - val_loss: 1.0910 - val_accuracy: 0.4200
5/5 [=====] - 0s 4ms/step - loss: 1.1349 - accuracy: 0.2867
5/5 [=====] - 0s 4ms/step
Epoch 1/10
22/22 [=====] - 0s 16ms/step - loss: 1.0790 - accuracy: 0.4000 - val_loss: 1.0916 - val_accuracy: 0.4067
Epoch 2/10
22/22 [=====] - 0s 10ms/step - loss: 1.0771 - accuracy: 0.4100 - val_loss: 1.0943 - val_accuracy: 0.4267
Epoch 3/10
22/22 [=====] - 0s 8ms/step - loss: 1.0747 - accuracy: 0.4057 - val_loss: 1.0897 - val_accuracy: 0.4333
Epoch 4/10
22/22 [=====] - 0s 7ms/step - loss: 1.0737 - accuracy: 0.4157 - val_loss: 1.0956 - val_accuracy: 0.4067
Epoch 5/10
22/22 [=====] - 0s 8ms/step - loss: 1.0733 - accuracy: 0.4186 - val_loss: 1.0923 - val_accuracy: 0.4200
Epoch 6/10
22/22 [=====] - 0s 7ms/step - loss: 1.0729 - accuracy: 0.4014 - val_loss: 1.0900 - val_accuracy: 0.4200
Epoch 7/10
22/22 [=====] - 0s 8ms/step - loss: 1.0712 - accuracy: 0.4114 - val_loss: 1.0909 - val_accuracy: 0.4133
Epoch 8/10
```

Fig 2

PREDICTED CLASS

```
File Edit View Insert Cell Kernel Help Trusted Python 3 (ipykernel) O
[Icons] [Run] [Code]
Epoch 7/10
22/22 [=====] - 0s 8ms/step - loss: 1.0712 - accuracy: 0.4114 - val_loss: 1.0909 - val_accuracy: 0.4133
Epoch 8/10
22/22 [=====] - 0s 11ms/step - loss: 1.0693 - accuracy: 0.4300 - val_loss: 1.0933 - val_accuracy: 0.3933
Epoch 9/10
22/22 [=====] - 0s 10ms/step - loss: 1.0689 - accuracy: 0.4300 - val_loss: 1.0889 - val_accuracy: 0.4000
Epoch 10/10
22/22 [=====] - 0s 10ms/step - loss: 1.0677 - accuracy: 0.4300 - val_loss: 1.0903 - val_accuracy: 0.4200
5/5 [=====] - 0s 4ms/step - loss: 1.1456 - accuracy: 0.3067
Accuracy on Test Set: 0.30666667222976685
5/5 [=====] - 0s 4ms/step
Predicted Class: 0, True Class: 1
Predicted Class: 0, True Class: 1
Predicted Class: 1, True Class: 1
Predicted Class: 1, True Class: 2
Predicted Class: 1, True Class: 1
```

Fig 3

CONCLUSION-

The problem statement emphasizes the importance of optimizing backpropagation in age classification using Python neural networks. Key factors include data quality, model architecture, hyperparameter tuning, custom loss functions, regularization techniques, and optimization algorithms. Solving these challenges will lead to improved age prediction models with applications in content filtering, marketing, and security. This project holds the potential to advance the field of computer vision and enhance age-related classification tasks.

REFERENCE-

- <https://numpy.org/doc/stable/>
- <https://www.tensorflow.org/>
- <https://en.wikipedia.org/wiki/Backpropagation>
- <https://jupyter.org/>

COURSE OUTCOME

CO6 : Program Genetic and Evolutionary algorithm.

Hence through this **PBL CO6** is Satisfied.