CSE-4132: Artificial Intelligence Lab

Md. Hasnain Ali
2010976153

1 Build a fully connected neural network (FCNN) and a convolutional neural network (CNN) for classifying 10 classes of images.

The fully connected neural network (FCNN) for 10-class classification is described below:

```
inputs = Input((28, 28, 1), name = 'InputLayer')
x = Flatten()(inputs)
x = Dense(1024, activation = 'relu')(x)
outputs = Dense(10, name = 'OutputLayer', activation = 'softmax')(x
    )
model = Model(inputs, outputs, name = 'Multi-Class-Classifier')
model.summary()
```

Listing 1: FCNN Model

The fully convolutional neural network (CNN) for 10-class classification is described below:

```
inputs = Input((28, 28, 1), name = 'InputLayer')
x = Conv2D(16, kernel_size=3,  activation="relu")(inputs)
x = MaxPooling2D()(x)
x = Conv2D(32, kernel_size=3, activation="relu")(x)
x = MaxPooling2D()(x)
x = Flatten()(x)
x = Dense(512, activation = 'relu')(x)
outputs = Dense(10, name = 'OutputLayer', activation = 'softmax')(x
    )
model = Model(inputs, outputs, name = 'Multi-Class-Classifier')
model.summary()
```

Listing 2: CNN Model

2 Train and test your FCNN and CNN by the Fashion dataset. Discuss your results by comparing performance between two types of networks.

We conducted a comparative analysis of two types of neural networks: the Fully Connected Neural Network (FCNN) and the Convolutional Neural Network (CNN), trained and tested on the Fashion MNIST dataset. The primary objective was to evaluate and discuss the performance differences between these architectures.

Hyperparameters

The hyperparameters for both FCNN and CNN models are summarized in Table 1.

Hyperparameter	**FCNN**	**CNN**
Input Dimensions	(28, 28, 1)	(28, 28, 1)
Number of Hidden Layers	1	2 (Convolutional) + 1 (Dense)
Number of Neurons/Filters	1024 neurons	16, 32 filters (Conv), 512 neurons (Dense)
Total Parameters	814090	420042
Activation Function	ReLU	ReLU
Output Layer	10 neurons (Softmax)	10 neurons (Softmax)
Batch Size	128	128
Epochs	10	10
Optimizer	RMSprop	RMSprop
Loss Function	Categorical Crossentropy	Categorical Crossentropy

Table 1: Hyperparameter Comparison of FCNN and CNN Models

Results

The models were evaluated on the test dataset using accuracy as the performance metric. The results are as follows:

- **FCNN Accuracy:** 88.76%
- **CNN Accuracy:** 89.88%

Analysis

Performance Comparison

The CNN outperformed the FCNN with an accuracy of 89.88%, compared to 88.76% achieved by the FCNN. The superior performance of the CNN can be

attributed to its ability to extract spatial features using convolutional layers, which is particularly beneficial for image data. In contrast, the FCNN treats each pixel independently, which limits its capacity to capture spatial relationships.

Model Complexity

The CNN architecture introduces additional complexity due to convolutional and pooling layers, which results in higher computational costs. However, this added complexity translates to better generalization and performance on the dataset.

Training Efficiency

Both models used similar training parameters, including the optimizer and batch size. Despite its higher complexity, the CNN demonstrated efficient training due to its hierarchical feature extraction mechanism.

The results demonstrate that CNNs are more effective than FCNNs for image classification tasks like Fashion MNIST, as they can capture spatial hierarchies and patterns in data. While FCNNs are simpler and computationally less demanding, they are less suited for image-based tasks where spatial relationships play a crucial role.

3 Build a CNN having a pre-trained MobileNet as the backbone to classify 10 classes

Answer: The CNN having a pre-trained MobileNe for 10-class classification is described below:

```
inputs_mobilenet = mobilenet.inputs
outputs_mobilenet = mobilenet.output
x = GlobalAveragePooling2D()(outputs_mobilenet)
x = Dense(1024, activation=None)(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
x = Dense(512, activation=None)(x)
x = BatchNormalization()(x)
outputs_final = Dense(10, name = 'OutputLayer', activation = '
    softmax')(x)
model = Model(inputs_mobilenet, outputs_final, name = 'Multi-Class-
    Classifier')
model.summary()
```

Listing 3: MobileNet for 10 Class Classification

4 Train and test your CNN having a pre-trained MobileNet as backbone to classify images of the CIFAR-10 dataset. Discuss your results by comparing performance between transfer learning + fine tuning and only transfer learning.

We analyzed classifying CIFAR-10 images using a pre-trained MobileNet model as the backbone. The two approaches, *Transfer Learning only* and *Transfer Learning + Fine Tuning*, are compared based on their accuracy.

Methodology

1. The CIFAR-10 dataset was preprocessed and normalized.
2. The MobileNet model was used as a feature extractor for Transfer Learning.
3. Additional fully connected layers were added on top of MobileNet for classification.
4. Experiments were conducted with two models:
   - **Transfer Learning only**: Freezing all MobileNet layers and training only the additional layers.
   - **Transfer Learning + Fine Tuning**: Fine-tuning some MobileNet layers along with the additional layers.

Results

The results of the experiments are summarized in Table 2. Fine-tuning significantly improves the classification accuracy.

Table 2: Comparison of Accuracy

Approach	**Accuracy**
Transfer Learning only	0.58
Transfer Learning + Fine Tuning	0.8004

Hyperparameter Comparison

Table 3 provides a comparison of the hyperparameters used in both approaches.

Table 3: Hyperparameter Comparison

Hyperparameter	**Transfer Learning only**	**Transfer Learning + Fine Tuning**
Learning Rate	0.001	0.0001
Batch Size	128	128
Epochs	50	50 (TL) + 30 (FT)
Trainable Layers	Final Dense Layers	Final Dense Layers + Last 7 Layers
Optimizer	Adam	Adam
Loss Function	Categorical Crossentropy	Categorical Crossentropy

Discussion

The results show a clear advantage of using Transfer Learning with Fine Tuning over Transfer Learning only. Fine-tuning allows the model to adapt pre-trained MobileNet weights to the CIFAR-10 dataset, which has distinct features compared to the ImageNet dataset used for pre-training.

The experiment demonstrates that Transfer Learning + Fine Tuning is a more effective approach for image classification tasks when compared to Transfer Learning only. Fine-tuning leverages the backbone model's pre-trained features and adapts them to the target dataset, leading to better performance.