BBM418 Assignment 3 Report

Mustafa Kemal Öz – 2230356179

Modeling and Training a CNN classifier from Scratch:

Model Design:

- The custom CNN model consists of 6 convolutional layers followed by max-pooling and 2 fully connected layers.
- ReLU activation function is used after each convolutional layer.
- Cross-Entropy Loss is used as the loss function, and Adam optimizer is used for optimization.

```python
class Model(nn.Module):
    def __init__(self, num_classes=10, dropout_prob=0.0):
        super(Model, self).__init__()
        self.conv1 = nn.Conv2d(3, 4, kernel_size=3, stride=1, padding=1)
        self.conv2 = nn.Conv2d(4, 8, kernel_size=3, stride=1, padding=1)
        self.conv3 = nn.Conv2d(8, 16, kernel_size=3, stride=1, padding=1)
        self.conv4 = nn.Conv2d(16, 32, kernel_size=3, stride=1, padding=1)
        self.conv5 = nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1)
        self.conv6 = nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.fc1 = nn.Linear(128 * 14 * 14, 512)
        self.fc2 = nn.Linear(512, num_classes)
        self.dropout = nn.Dropout(p=dropout_prob)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = F.relu(self.conv3(x))
        x = self.pool(F.relu(self.conv4(x)))
        x = F.relu(self.conv5(x))
        x = self.pool(F.relu(self.conv6(x)))

        x = torch.flatten(x, 1)

        x = F.relu(self.fc1(x))
        x = self.dropout(x)
        x = self.fc2(x)
        return x
```

Figure 1: CNN Classifier

Training and Evaluation:

The model is trained for 100 epochs with different batch sizes and learning rates.

- Learning Rate: 0.01, Batch Size: 30
- Validation Accuracy: 10%, Test Accuracy: 10%, Loss: 2.305
- Learning Rate: 0.001, Batch Size: 30
- Validation Accuracy: 32%, Test Accuracy: 26%, Loss: 7.00e-06
- Learning Rate: 0.0001, Batch Size: 30
- Validation Accuracy: 34.5%, Test Accuracy: 35%, Loss: 0.00114
- Learning Rate: 0.01, Batch Size: 60
- Validation Accuracy: 10%, Test Accuracy: 10%, Loss: 2.304
- Learning Rate: 0.001, Batch Size: 60

- Validation Accuracy: 26%, Test Accuracy: 21.5%, Loss: 1.15e-05
- Learning Rate: 0.0001, Batch Size: 60
- Validation Accuracy: 26%, Test Accuracy: 28.5%, Loss: 0.00218
- Learning Rate: 0.01, Batch Size: 90
- Validation Accuracy: 10%, Test Accuracy: 10%, Loss: 2.303
- Learning Rate: 0.001, Batch Size: 90
- Validation Accuracy: 30%, Test Accuracy: 28.5%, Loss: 9.59e-08
- Learning Rate: 0.0001, Batch Size: 90
- Validation Accuracy: 31.5%, Test Accuracy: 30.5%, Loss: 0.010

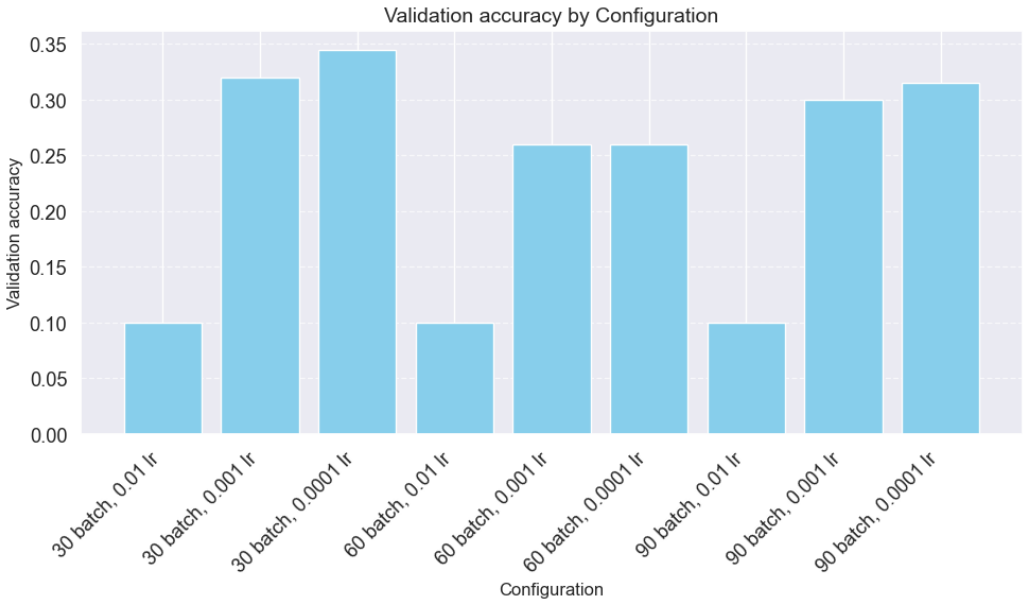

Figure 2: Validation Accuracy Graph

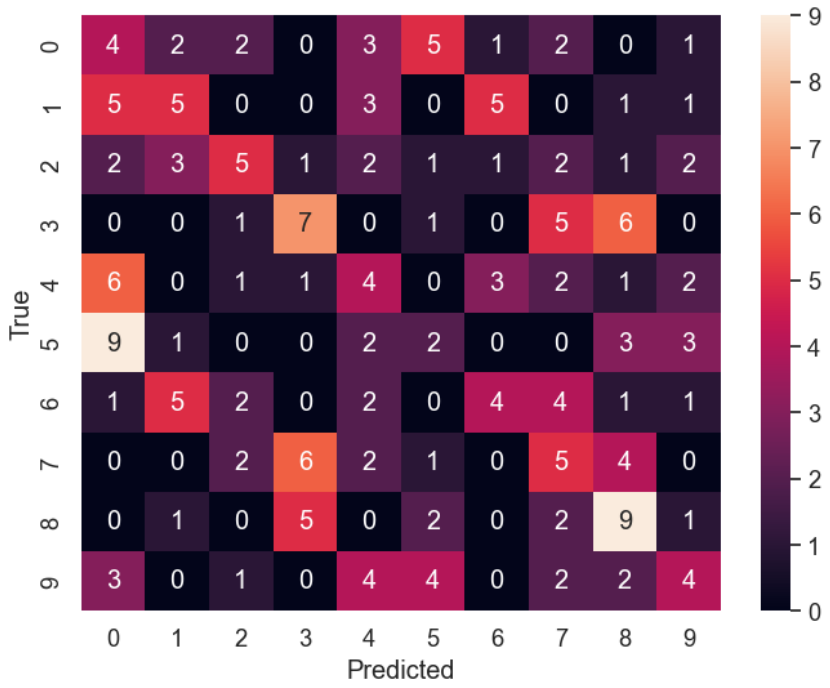

Figure 3: Best Model Confusion Matrix

Dropout:

Different dropout values are tested with the best model:

- Dropout: 0.1, Validation Accuracy: 33%, Test Accuracy: 27%, Loss: 0.012
- Dropout: 0.3, Validation Accuracy: 22.5%, Test Accuracy: 24%, Loss: 0.039
- Dropout: 0.6, Validation Accuracy: 24%, Test Accuracy: 25%, Loss: 0.155
- Dropout: 0.9, Validation Accuracy: 27%, Test Accuracy: 24.5%, Loss: 0.753

Transfer Learning with EfficientNet:

Fine-tuning Explanation:

Fine-tuning is a process in deep learning where a pre-trained neural network model, initially trained on a large dataset like ImageNet, is adapted to a new task or dataset. This involves reusing the knowledge gained by the pre-trained model and making adjustments to its top layers while keeping the lower layers frozen. By modifying the top layers to match the new task's requirements and fine-tuning the model on the new dataset, it can learn task-specific features while retaining the general knowledge acquired during pre-training.

- EfficientNet-B0 model is fine-tuned on the given dataset by freezing all layers except the fully connected (FC) layer.
- The last layer of the network is modified to adapt to the new classification task.

Training with EfficientNet:

- Two cases of fine-tuning are explored: training only the FC layer and freezing the rest, and training the last two convolutional layers and FC layer while freezing the rest.
- The models are trained with a learning rate of 0.0001 for 100 epochs.

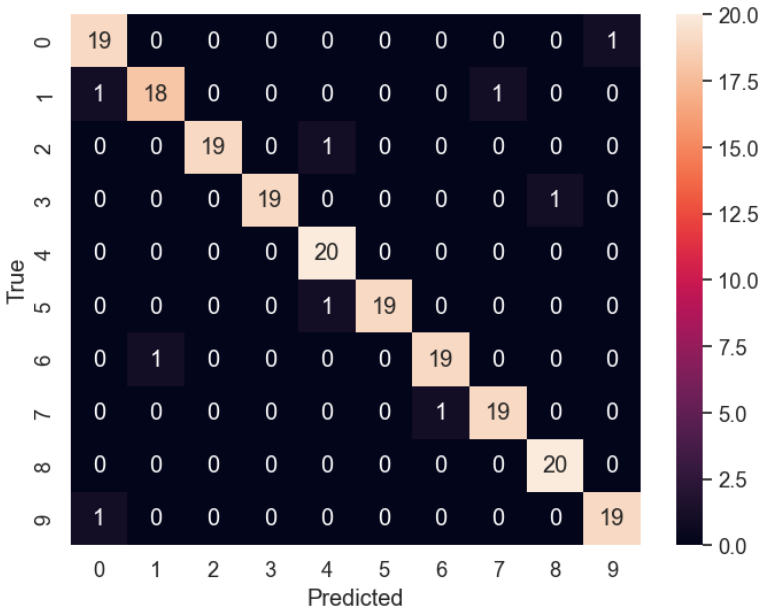

Figure 4: Fine Tuned EfficientNet Confusion Matrix

Observations:

- The custom CNN model achieves varying accuracy with different configurations of batch size and learning rate.
- Dropout improves the performance of the best CNN model.
- Fine-tuning the EfficientNet model yields competitive results compared to the custom CNN model.
- Two versions of the fine tuned models that specified in this assignment has performed similar performances however the first one performed sligthly better than the second model.