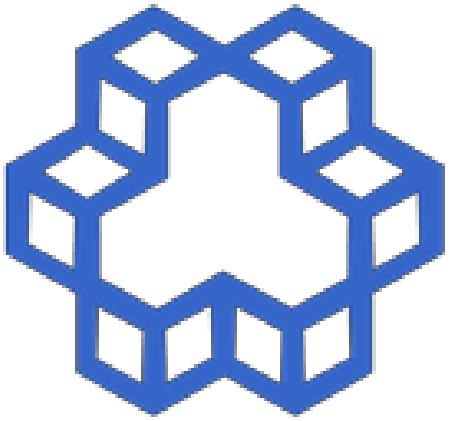

K. N. Toosi
University of Technology

Clothing Classification

Introduction:

In this project, our goal is to design and implement Convolutional Neural Network (CNN) models for classifying clothing images. The dataset used consists of 70,000 images of clothing, with 60,000 images set aside for training and validation, and 10,000 images reserved for testing.

Challenges:

1. **Importing the necessary libraries.**
2. **Designing three convolutional models with different architectures.**
3. **Evaluating the models and comparing them based on accuracy.**

Results of the Models and Their Comparison:

First, let's take a look at the hyperparameters that I considered for each of them:

```
history1 = model1.fit(X_train, y_train, validation_split=0.2, epochs=5, batch_size=128)
```

```
history2 = model2.fit(X_train, y_train, validation_split=0.2, epochs=5, batch_size=128)
```

```
history3 = model3.fit(X_train, y_train, validation_split=0.2, epochs=5, batch_size=128)
```

And then I included the chart for each model along with an analysis:

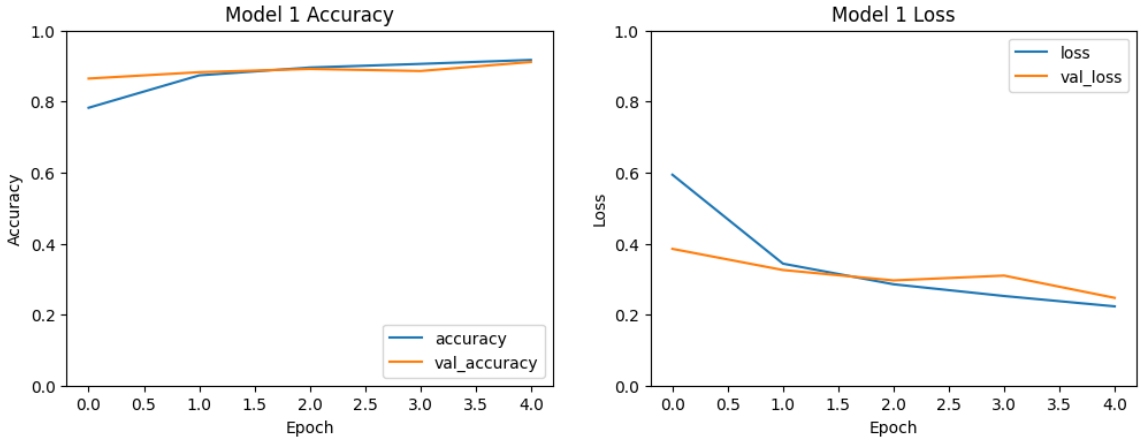

Figure 1: Epoch-Accuracy and Epoch-Loss Chart

Accuracy and Loss Charts for Model 1

Model 1 Accuracy:

- This chart shows that the model's accuracy has improved over the training epochs.
- The accuracy for training and validation data is nearly similar and increasing, indicating the model's good performance in learning.
- The training accuracy increased from approximately 0.84 to 0.92 over the 5 epochs.
- The validation accuracy increased from about 0.87 to 0.91 over the 5 epochs.

Model 1 Loss:

- The model's loss has also decreased over the training epochs.
- A similar decrease in validation loss is observed, indicating a reduction in error in the model's predictions.
- The training loss decreased from approximately 0.5 to 0.2.

- The validation loss decreased from around 0.4 to 0.3.

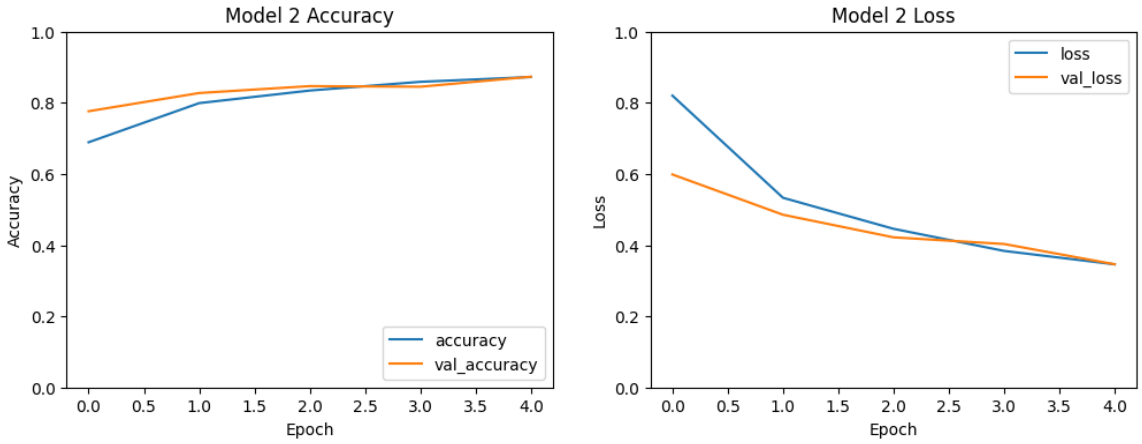

Figure 2: Epoch-Accuracy and Epoch-Loss Chart

Accuracy and Loss Charts for Model 2

Model 2 Accuracy:

- The accuracy of Model 2 has also improved over the training epochs.
- The training and validation accuracy are nearly similar and show comparable improvements, indicating a good balance between learning and validation.
- The training accuracy increased from approximately 0.82 to 0.91 over the 5 epochs.
- The validation accuracy increased from about 0.85 to 0.90 over the 5 epochs.

Model 2 Loss:

- The loss for Model 2 has also decreased over the training epochs.
- The validation data shows a similar decrease in loss, indicating improvements in the model's predictions.

- The training loss decreased from approximately 0.6 to 0.3.
- The validation loss decreased from around 0.5 to 0.4.

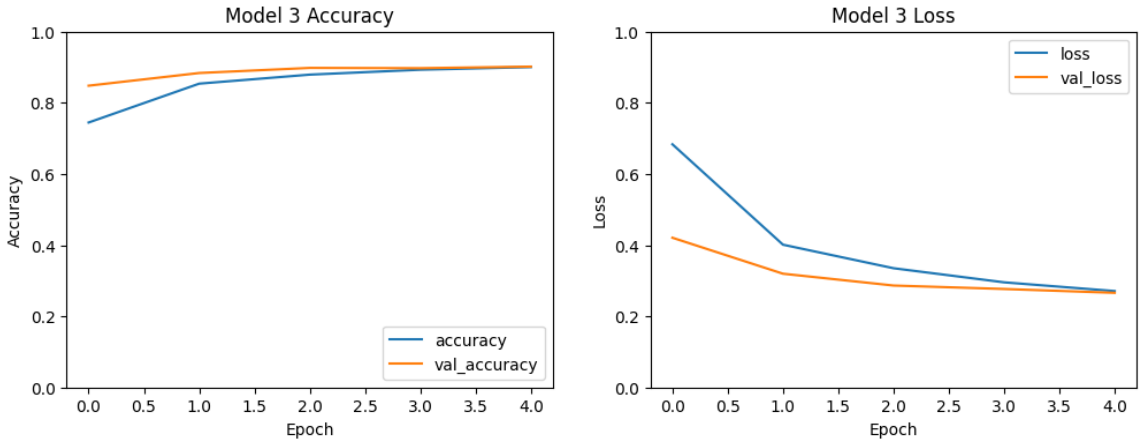

Figure 3: Epoch-Accuracy and Epoch-Loss Chart

Accuracy and Loss Charts for Model 3

Model 3 Accuracy:

- The accuracy of Model 3 has also improved over the training epochs.
- The training and validation accuracy are nearly similar and show comparable improvements, indicating a good balance between learning and validation.
- The training accuracy increased from approximately 0.86 to 0.92 over the 5 epochs.
- The validation accuracy increased from about 0.88 to 0.92 over the 5 epochs.

Model 3 Loss:

- The loss for Model 3 has also decreased over the training epochs.
- The validation data shows a similar decrease in loss, indicating improvements in the model's predictions.

- The training loss decreased from approximately 0.4 to 0.2.
- The validation loss decreased from around 0.3 to 0.2.

Final Results of the Models' Accuracy:

- Model 1 Test Accuracy: 0.9135
- Model 2 Test Accuracy: 0.8783
- Model 3 Test Accuracy: 0.9022

Based on the results obtained, Model 1 has shown the best performance. Model 3, with the addition of a Dropout layer, has slightly reduced the accuracy compared to Model 1.

Model 1:

- It has a good final accuracy (0.9135), and the model's error has consistently decreased. The accuracy and loss charts indicate that the model has been well trained and generalized effectively.

Model 2:

- The final accuracy (0.8783) is lower than the other two models. This model may perform better for certain types of data due to the use of Average Pooling, but it did not show better performance in this case.

Model 3:

- With the addition of a Dropout layer, the final accuracy has improved (0.9022), and it seems that overfitting has been reduced. This model performs better than Model 2 but still lags behind Model 1.

Now we will proceed to analyze the models as follows:

Model 1:

This model has four convolutional layers, each followed by a MaxPooling layer.

```
model1 = Sequential([
    Input(shape=(28, 28, 1)),
    Conv2D(32, (3, 3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, (3, 3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(128, (3, 3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(256, (3, 3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

Analysis of Model 1:

1. **Convolutional Layers:**
    - The number of filters is 32, 64, 128, and 256, respectively.
    - The filter size is (3, 3), and the ReLU activation function is used.
2. **Pooling Layers:**
    - MaxPooling with a size of (2, 2) is used, which reduces the dimensions of the features.
3. **Final Layers:**
    - A Flatten layer to convert multi-dimensional data to one-dimensional.

- A Dense layer with 128 neurons and a ReLU activation function.
- An output Dense layer with 10 neurons and a softmax activation function for final classification.

Model 2:

This model also has four convolutional layers but uses AveragePooling instead of MaxPooling.

```
model2 = Sequential([
    Input(shape=(28, 28, 1)),
    Conv2D(32, (3, 3), activation='relu', padding='same'),
    AveragePooling2D(pool_size=(2, 2)),
    Conv2D(64, (3, 3), activation='relu', padding='same'),
    AveragePooling2D(pool_size=(2, 2)),
    Conv2D(128, (3, 3), activation='relu', padding='same'),
    AveragePooling2D(pool_size=(2, 2)),
    Conv2D(256, (3, 3), activation='relu', padding='same'),
    AveragePooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

Analysis of Model 2:

1. **Convolutional Layers:**
   - The number of filters and the filter sizes are the same as in Model 1.
2. **Pooling Layers:**
   - AveragePooling with a size of (2, 2) is used, which takes the average of the values instead of the maximum.
3. **Final Layers:**

- Similar to Model 1.

Model 3:

This model is based on the best model between Model 1 and Model 2 and includes a Dropout layer to prevent overfitting.

```
model3 = Sequential([
    Input(shape=(28, 28, 1)),
    Conv2D(32, (3, 3), activation='relu', padding='same'),
       MaxPooling2D(pool_size=(2,  2))  if  best_model  ==  model1  else
AveragePooling2D(pool_size=(2, 2)),
    Conv2D(64, (3, 3), activation='relu', padding='same'),
       MaxPooling2D(pool_size=(2,  2))  if  best_model  ==  model1  else
AveragePooling2D(pool_size=(2, 2)),
    Conv2D(128, (3, 3), activation='relu', padding='same'),
       MaxPooling2D(pool_size=(2,  2))  if  best_model  ==  model1  else
AveragePooling2D(pool_size=(2, 2)),
    Conv2D(256, (3, 3), activation='relu', padding='same'),
       MaxPooling2D(pool_size=(2,  2))  if  best_model  ==  model1  else
AveragePooling2D(pool_size=(2, 2)),
    Flatten(),
    Dropout(0.5),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

Analysis of Model 3:

1. **Convolutional and Pooling Layers:**
    - These are the same as in the selected best model (either Model 1 or Model 2).
2. **Dropout Layer:**
    - A Dropout layer is added with a rate of 0.5 to reduce overfitting.
3. **Final Layers:**
    - Similar to Models 1 and 2.

Now, we will slightly change the hyperparameters to reevaluate the results. The hyperparameters are as follows:

```
history1 = model1.fit(X_train, y_train, validation_split=0.2, epochs=10, batch_size=128)
```

```
history2 = model2.fit(X_train, y_train, validation_split=0.2, epochs=10, batch_size=128)
```

```
history3 = model3.fit(X_train, y_train, validation_split=0.2, epochs=10, batch_size=128)
```

And then I have included the chart for each model along with the analysis below:

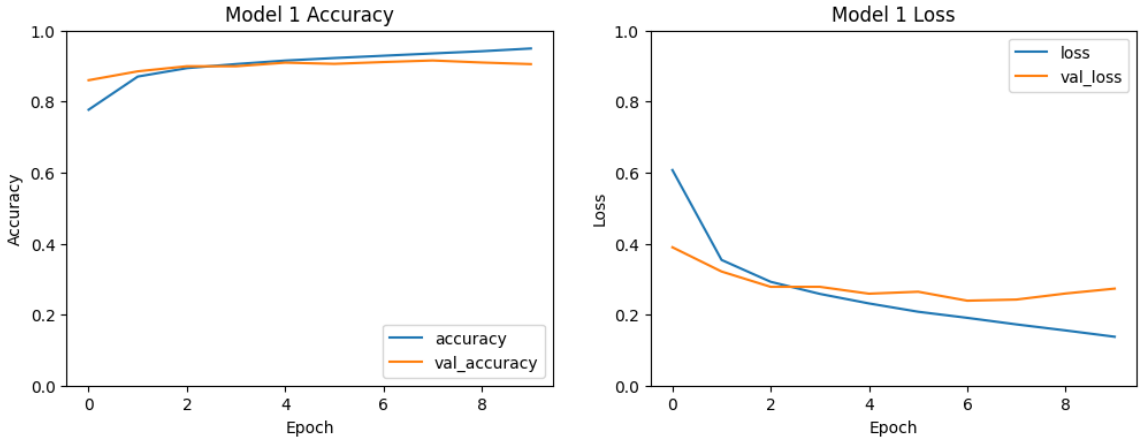

Figure 4: Epoch-Accuracy and Epoch-Loss Chart

Model 1:

- **Model 1 Test Accuracy:** 0.9129

Model 1 Accuracy:

- At the beginning of training, the model's accuracy in the first epoch is about 0.84.
- The model's accuracy increases to about 0.88 in the second epoch.
- By the end of the ninth epoch, the model's accuracy reaches 0.91.

Model 1 Loss:

- At the beginning of training, the model's loss in the first epoch is about 0.5.
- In the third epoch, the loss decreases to about 0.3.
- By the end of the ninth epoch, the loss decreases to about 0.2.

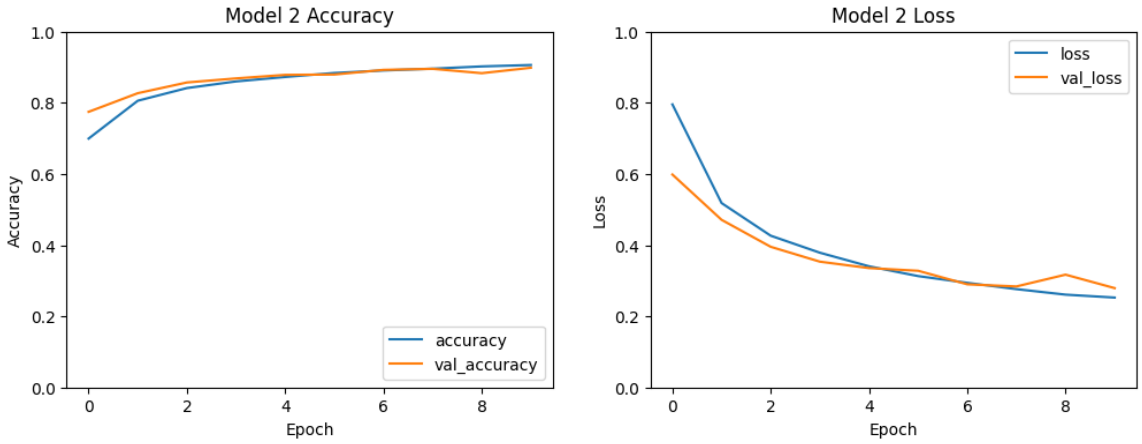

Figure 5: Epoch-Accuracy and Epoch-Loss Chart

Model 2:

- **Model 2 Test Accuracy:** 0.8980

Model 2 Accuracy:

- At the beginning of training, the model's accuracy in the first epoch is about 0.82.
- The model's accuracy increases to about 0.86 in the second epoch.
- By the end of the ninth epoch, the model's accuracy reaches 0.90.

Model 2 Loss:

- At the beginning of training, the model's loss in the first epoch is about 0.6.
- In the third epoch, the loss decreases to about 0.4.
- By the end of the ninth epoch, the loss decreases to about 0.3.

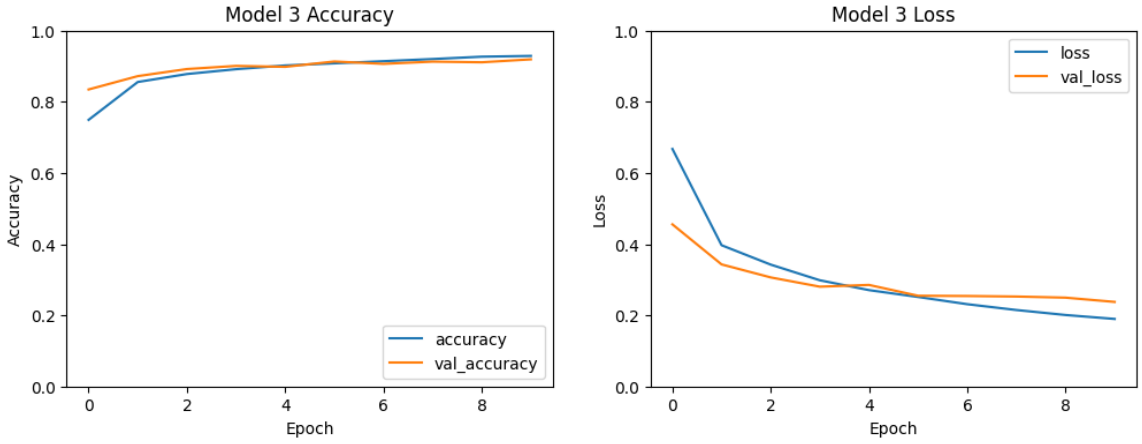

Figure 6: Epoch-Accuracy and Epoch-Loss Chart

Model 3:

- **Model 3 Test Accuracy:** 0.9217

Model 3 Accuracy:

- At the beginning of training, the model's accuracy in the first epoch is about 0.86.
- The model's accuracy increases to about 0.90 in the second epoch.
- By the end of the ninth epoch, the model's accuracy reaches 0.92.

Model 3 Loss:

- At the beginning of training, the model's loss in the first epoch is about 0.4.
- In the third epoch, the loss decreases to about 0.3.
- By the end of the ninth epoch, the loss decreases to about 0.2.

Final Results of Model Accuracies:

- Model 1 Test Accuracy: 0.9129
- Model 2 Test Accuracy: 0.8980
- Model 3 Test Accuracy: 0.9217

Based on the test accuracy, Model 3 (Model 3) has performed the best with a test accuracy of 0.9217. The addition of the Dropout layer has helped reduce overfitting and increased the model's accuracy.

The analysis of the models in this case is similar to the previous case.