INFOMCV Assignment 4

Authors: Sotiris Zenios, Andreas Alexandrou (Group 23)

Description and motivation of your baseline model and four variants

(For your baseline model, add a description and motivation of the architecture and parameters (which layers, which dimensions, how connected, etc.). Also use model.summary(). For each of the four variants, add a description of which property differs from the baseline model, and why this choice was made. Make sure you name/number your models so you can refer to them. Approx. 0.5-1 page.)

1.Baseline

The LeNet-5 architecture is a CNN for classifying 28x28 grayscale images. It begins with a convolutional layer (Conv2d-1) using 6 filters of size 5x5 and padding of 2 resulting in an output shape of 28x28 with 6 channels. The next convolutional layer (Conv2d-2) increases the depth to 16 channels using 5x5 filters, reducing the spatial dimensions to 10x10 focusing on more complex features. Following convolutional layers, the network flattens the output and transitions to fully connected layers, where Linear-3 transforms the input to 120 features, Linear-4 compresses it to 84 features and finally Linear-5 maps it to the 10 output classes, corresponding to the dataset categories. The architecture uses ReLU activation for non-linearity and max pooling to reduce dimensions and computational load with a total of 61,706 parameters.

2.3x3 Kernel

The variant LeNet5_3x3 changes the kernel size in both convolutional layers from 5x5 to 3x3. This change aimed to provide a more detailed feature extraction by focusing on smaller regions of the input image.The padding in the first convolutional layer is adjusted to 1 to maintain the output size. Despite the smaller kernel size, the total number of parameters increases to 81,194 due to the adjusted fully connected layer that now processes a larger feature map output from the second convolutional layer (12x12 instead of 10x10).

3. Leaky ReLu - 3x3 Kernel

The LeNet5_ActivationVariant substitutes ReLU with Leaky ReLU in its architecture to combat the "dying ReLU" issue, ensuring continuous learning even for neurons that output negative values. This model, carrying 81,194 parameters, keeps the convolutional layers' 3x3 kernel size and applies Leaky ReLU across all layers, aiming to model generalization on dataset.

4. Dropout - Leaky ReLu - 3x3 Kernel

The LeNet5ModifiedWithDropout variant integrates dropout regularization into the previous architecture to prevent overfitting. This model maintains the initial structure but incorporates dropout layers with a 50% dropout rate after each of the first two fully connected layers. The dropout layers randomly zero half of the activations during training, preventing reliance on any small set of neurons and therefore enhancing the model's generalization capabilities. Despite the addition of dropout, the total parameter count remains at 81,194 parameters, as dropout does not introduce additional parameters .

5. Avg Pool - Dropout - Leaky ReLu - 3x3 Kernel

The new variant, LeNet5_AvgPooling, modifies the previous architecture by replacing max pooling with average pooling. This change shifts the model's approach to downsampling feature maps in convolutional layers from selecting the maximum value within a pooling window to computing the average. Average pooling tends to smooth out the feature maps leading to a more generalized feature representation by capturing the average of features within a pooling. Rest of the model

structure, including convolutional and fully connected layers, remains unchanged, maintaining the original parameter count and computational complexity.

Training and validation loss for all five models

(Put, on each row, the loss and accuracy graphs side-by-side. The x-axis should have the epochs, the y-axis loss/accuracy. Uses lines with a different color for training and validation. Approx. 1 page)

Figure 1 Baseline Model

Figure 2 3x3 Kernel Model

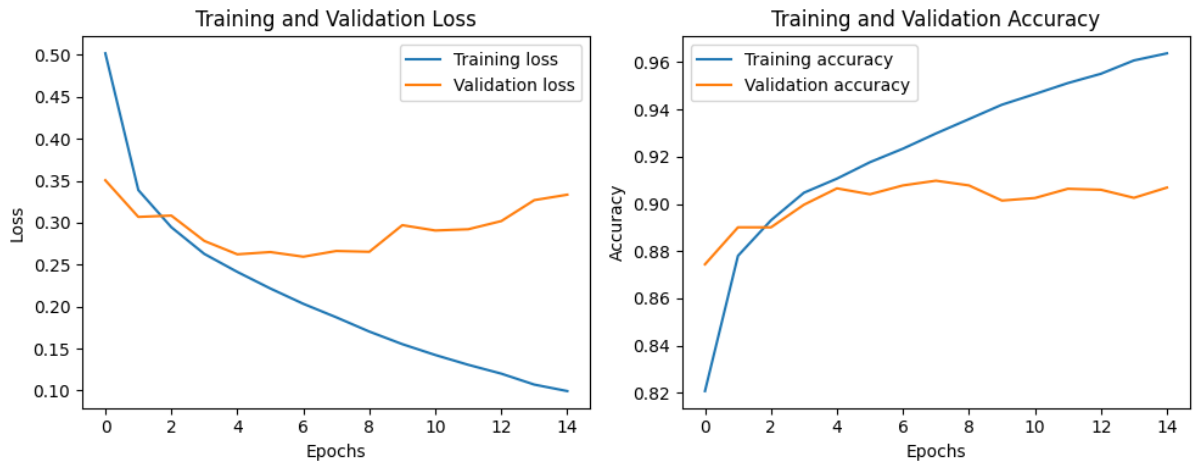

Figure 3 3x3 Kernel, Leaky ReLu Model

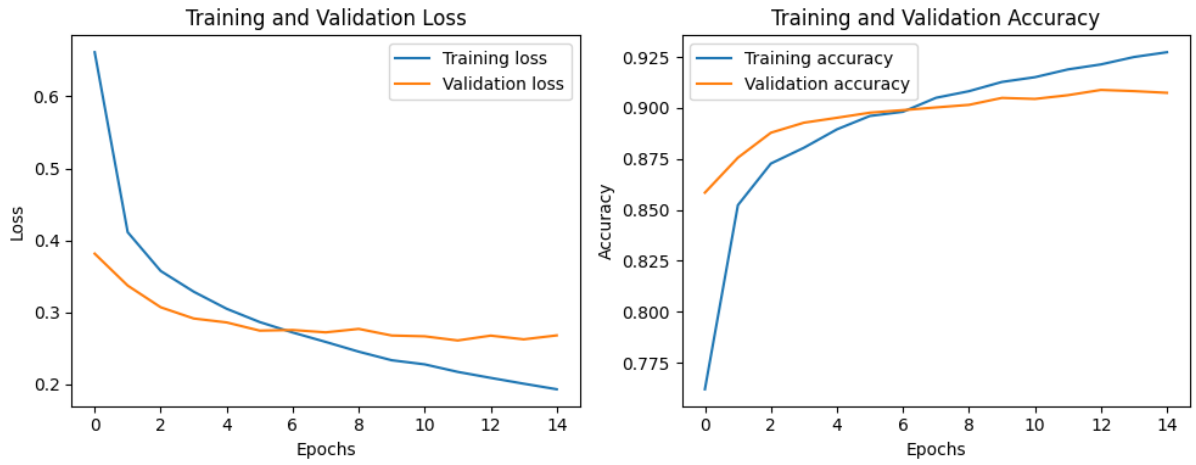

Figure 4 Dropout - Leaky ReLu - 3x3 Kernel

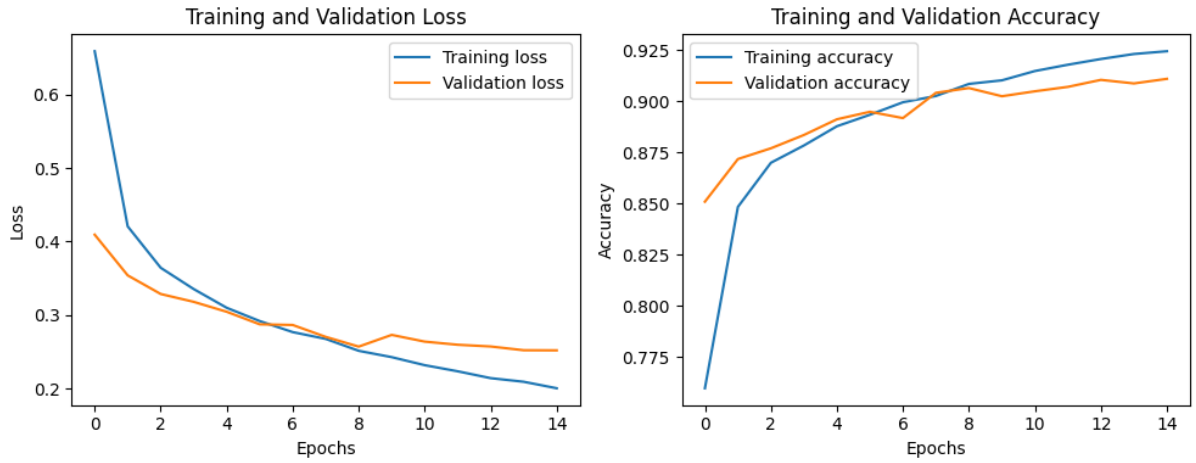

Figure 5 Avg Pool - Dropout - Leaky ReLu - 3x3 Kernel

Link to your model weights

(Link should be accessible by Ronald Poppe and Metehan Doyran.)

model_weights

Table with training and validation top-1 accuracy for all five models

(Fill the table below.)

Model name	Training top-1 accuracy (%)	Validation top-1 accuracy (%)
1.Baseline	95.08%	89.60%
2.3x3 Kernel	96.75%	89.95%
3.Leaky ReLu - 3x3 Kernel	96.37%	90.69%
4. Dropout-Leaky ReLu - 3x3 Kernel	92.73%	90.74%
5. Avg Pool - Dropout - Leaky ReLu - 3x3 Kernel	92.43%	91.09%

Discuss your results in terms of your model

(Discuss the results in terms of complexity, type of layers, overfitting measures, etc. Make pair-wise comparisons between the four variants and the baseline model. Approx. 0.5 page.)

Analyzing the results of the five model variants we can make the following observations.

(Models 1 to 2): Transitioning from the baseline model to the 3x3 kernel variant led to an increase in both training and validation accuracies. The smaller kernel size in Model 2 allows for more detailed feature extraction.

(Models 2 to 3): Introducing Leaky ReLU in Model 3, while keeping the 3x3 kernel size, slightly improved validation accuracy but with a marginal decrease in training accuracy. Leaky ReLU helps combat the vanishing gradient problem by allowing a small gradient when the unit is inactive, leading to more effective learning during training. The higher validation accuracy shows better generalization, due to the more nuanced activation offered by Leaky ReLU.

(Models 3 to 4): The addition of dropout in Model 4 resulted in a decrease in training accuracy but an increase in validation accuracy. Dropout is a form of regularization, deliberately "forgetting" random parts of information during training to prevent the model from overfitting. The drop in training performance is expected due to the regularization effect, but the increase in validation accuracy indicates enhanced generalization to unseen data.

(Models 4 to 5): Replacing max pooling with average pooling in Model 5 led to a slight decrease in training accuracy but the highest validation accuracy among all models. Average pooling aggregates the feature map information by computing the average, which lead to smoother and more generalizable feature representations. This explains the model's improved performance on the validation set showing that average pooling contributes positively to the model's ability to generalize.

Discuss the differences between the two models evaluated on the test set
(Discuss the potential causes for (lack of) differences in terms of architecture. Also compare the results of each model to the training performance. Approx. 0.5 page.)

In evaluating the performance of two models (our best performing model) on the Fashion MNIST dataset, one trained on the training set and the other on a combination of the training and validation sets, we observed few differences in their performances. The model trained exclusively on the training dataset achieved a training accuracy of 92.43% and a test accuracy of 91.09%, while the model trained on both the training and validation datasets showed a slightly higher training accuracy of 92.8% and a test accuracy of 91.25%.

The marginal improvement in test accuracy for the model trained on the combined dataset suggests that the additional examples from the validation set contributed positively towards the model's ability to generalize. This improvement, even though it is small, indicates that the extra data provided more diverse examples or learning patterns beneficial for test set performance. The very slight difference in training accuracies (92.43% vs. 92.8%) also supports the idea that the additional data had a modest effect on the model's learning capability.

The minimal difference in performance could be attributed to several factors, including the architecture of the LeNet5Modified model and the characteristics of the Fashion MNIST dataset. Firstly, the LeNet5Modified architecture, being relatively simple, might be close to optimal for the task with the given dataset size, leaving limited room for improvement through additional training data alone. The architecture's capacity might be sufficiently tuned to the complexity of the task, meaning that beyond a certain point, more data does not equate to better performance. Furthermore, the Fashion MNIST dataset is diverse and homogenous in terms of the types of images and the variations within classes. The additional examples from the validation set may not have introduced significantly new information to significantly change the model's learning.

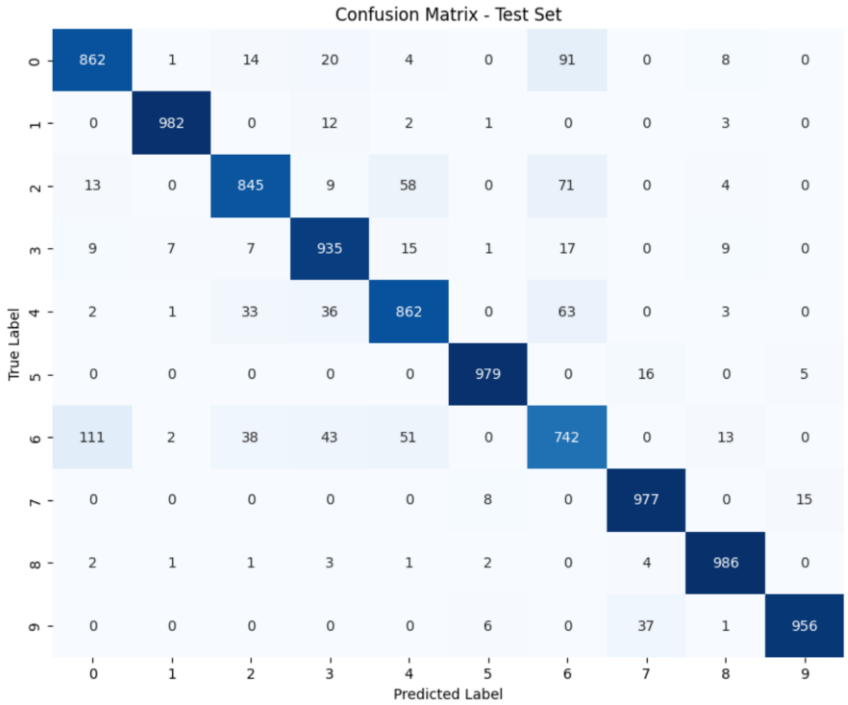

Figure 6: Confusion matrix (train+val model) on test set

Choice tasks

(Indicate which ones you did, and how you did them; Approx. half a page.)

CHOICE 2: k-fold cross-validation

In the task of implementing k-fold cross-validation to evaluate our model's performance on the Fashion MNIST dataset, we applied a 5-fold cross-validation strategy. This technique involved partitioning the dataset into five distinct folds, subsequently training and validating the model five separate times. Each iteration utilized a different fold as the validation set while combining the remaining four folds for training, ensuring that each fold served as the validation set exactly once. The process repeats for all folds, resulting in an average accuracy across folds. This method is used to prevent overfitting by validating the model on multiple, diverse subsets of the data. This cross-validation approach offered a more comprehensive assessment of the model's performance, reducing the dependency on a singular fixed validation set and providing insights into the model's stability across different data subsets. We used the k fold cross-validation on our baseline model and the validation accuracy increased from 89.60% to an average validation accuracy of 90.19%. This is because we allowed the model to train and validate across multiple distinct sets, ensuring a more comprehensive learning and evaluation process that better generalizes across the entire dataset.

CHOICE 4: Perform data augmentation techniques.

In order to boost the performance of our model we implemented data augmentation techniques. These techniques include random horizontal flip, to be able to generalize by simulating the scenario where clothing items can be oriented in different direction and random rotation up to 20 degrees, to mirror real world situation where the clothing is not perfectly aligned. The reason of these techniques is to let our model learn a more robust representation of the item even if it’s not perfectly positioned. Furthermore, we adjusted brightness and contrast to help the model be more robust in lighting variations and Gaussian blur that introduces a level of defocus that can occur in some pictures. With this data augmentation technique, we were able to keep the test accuracy high with around 89% accuracy. That is a bit less than the performance of our best model but is expected to perform better with real life pictures of the said objects in the dataset.

CHOICE 5: Provide t-SNE visualization of the fully connected layer before your output layer.

To better understand our model's classification behaviour, we applied t-SNE dimensionality reduction technique to the high dimensional embedding from our LeNet-5 model's fully connected layer before your output layer (fc2). From the visualization we can understand that several classes have form distinct and separated clusters, which shows that the model discern these classes with a high degree of confidence (e.g. class 9, class 8). Notably, there are regions where classes appear to overlap. Those are areas where the model had class confusions, like for example class 0 with class 6 where a T-shirt and a Shirt are easier to get confused and misclassified by the model. Some outliers form the visualization might represent misclassifications, especially if these outliers are closer to clusters of other classes.

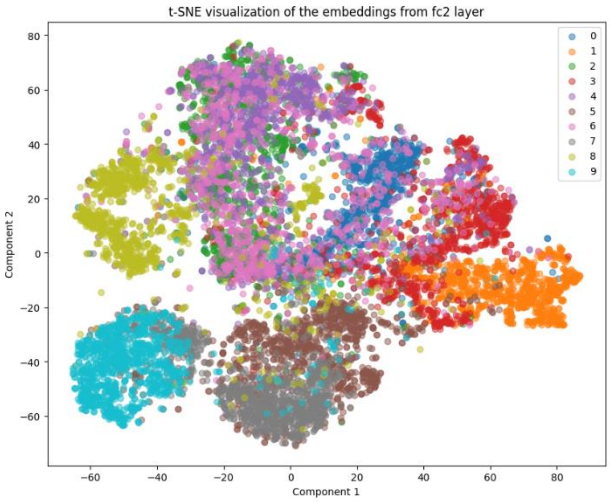

Figure 7 t-SNE visualisation