University of St.Gallen

School of Computer Science (SCS)
Artificial Intelligence & Machine Learning [AI:ML]

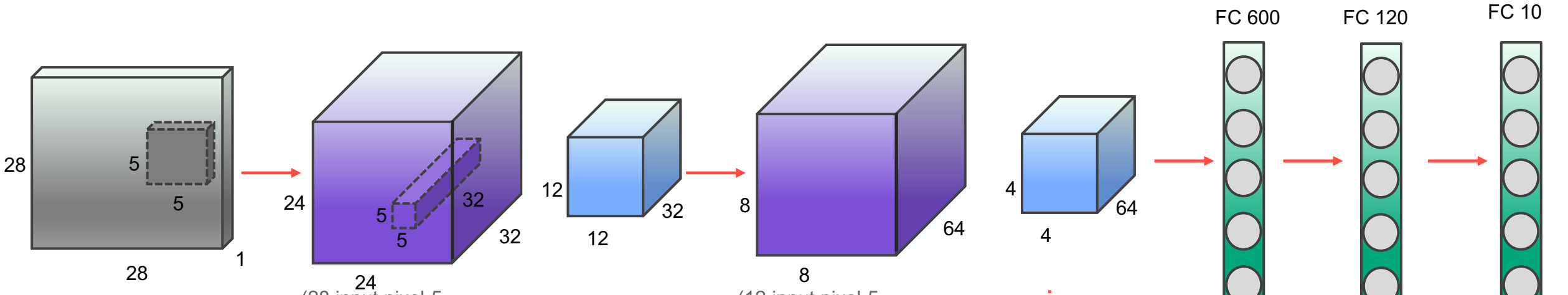
10,831,1.00 Deep Learning: Fundamentals and Applications

Fashion MNIST Convolutional Neural Networks

Kayla Kahn

Artificial Intelligence and Machine Learning
University of St.Gallen

CNN Architecture – Basic Model



Input: 1x28x28 images

(28 input pixel-5 kernel pixel)+1=24

Conv layer 1:
Expects 1 input channel, will convolve 32 filters of size 1x5x5 (stride=1 and no zero padding)

Output size becomes 32x24x24

Relu activation

Max Pool

Size becomes 32x12x12

(12 input pixel-5 kernel pixel)+1=8

Conv layer 2:
Expects 32 input channels, will convolve 64 filters of size 32x5x5

Output size becomes 64x8x8

Relu activation

Max Pool

Size becomes 64x4x4

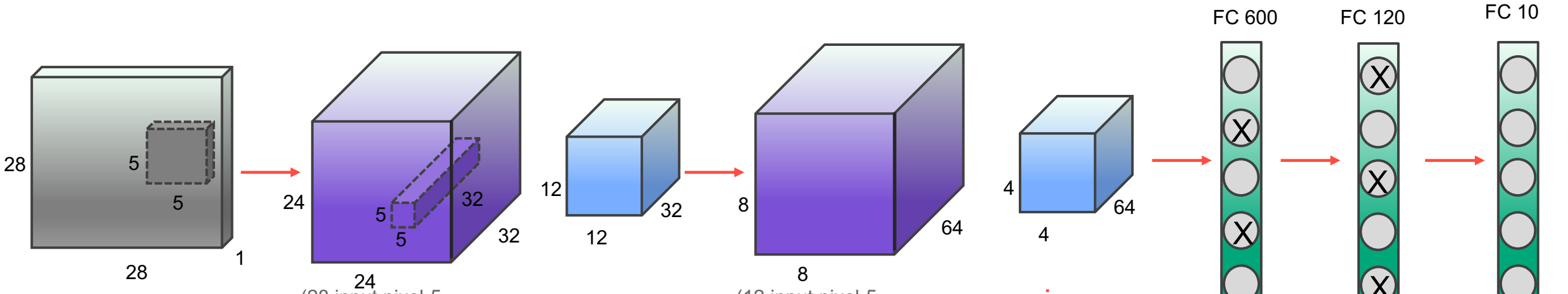
Fully connected layers

Relu activation

Softmax fully connected layer with 10 classes for the final classification

Artificial Intelligence and Machine Learning
University of St.Gallen

CNN Architecture – Better Model



Input: 1x28x28 images

(28 input pixel-5 kernel pixel)+1=24

Conv layer 1:
Expects 1 input channel, will convolve 32 filters of size 1x5x5 (stride=1 and no zero padding)

Output size becomes 32x24x24

Relu activation

Max Pool

Size becomes 32x12x12

(12 input pixel-5 kernel pixel)+1=8

Conv layer 2:
Expects 32 input channels, will convolve 64 filters of size 32x5x5

Output size becomes 64x8x8

Relu activation

Max Pool

Size becomes 64x4x4

Fully connected layers ***with dropout regularization with a neuron de-activation probability of 50% in each of the two layers***

Relu activation

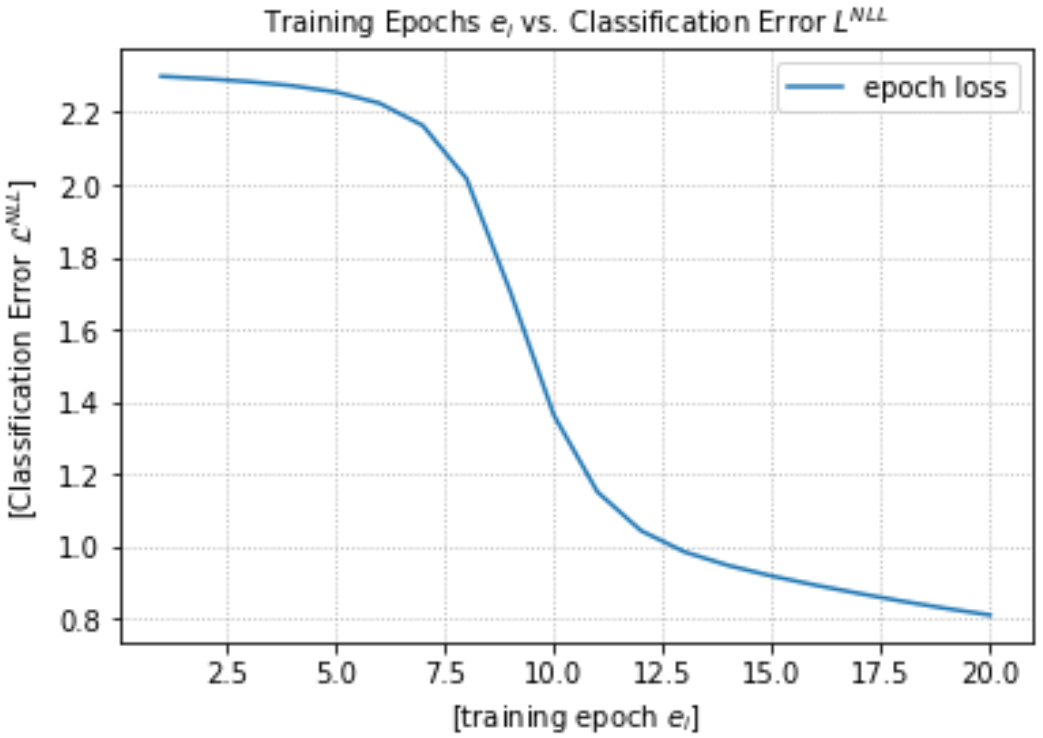
Softmax fully connected layer with 10 classes for the final classification

CNN Models – Overall Comparison

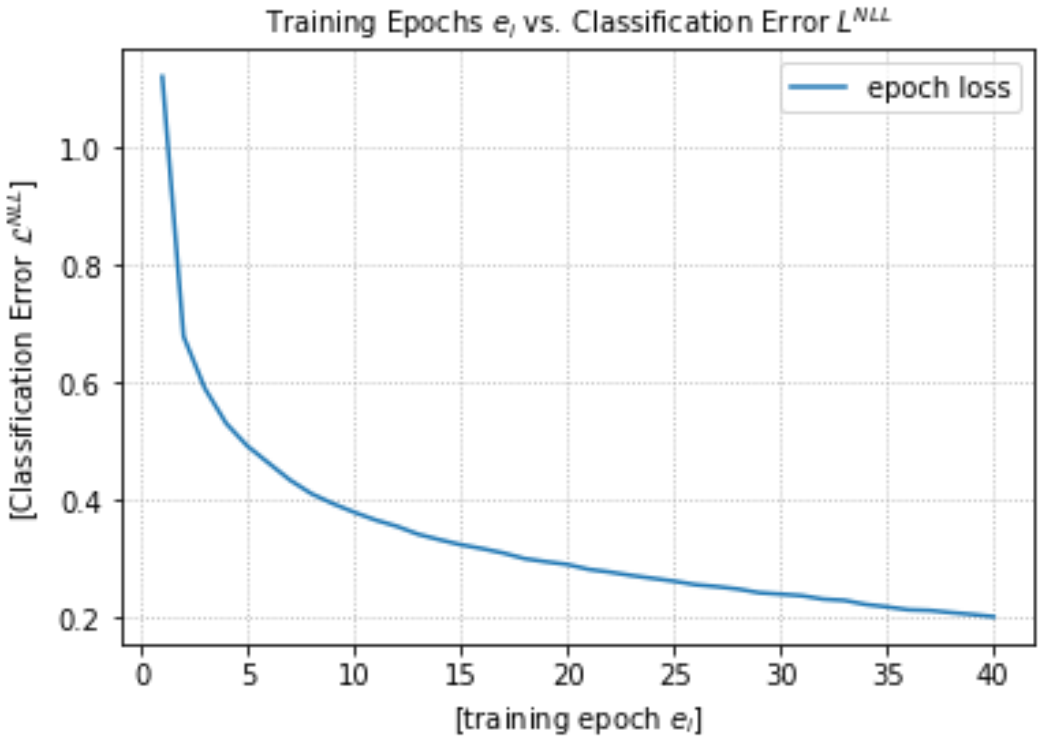
	Baseline Model	Better Model
Architectural Differences	no regularization	dropout regularization on two fully connected layers (but not on last layer)
Optimizer	Stochastic gradient descent	Adam
Learning rate	0.001	0.0001 (initially, but updated by Adam optimizer)
Epochs	20	40
Loss	negative log likelihood	negative log likelihood

Training Performance

Baseline Model



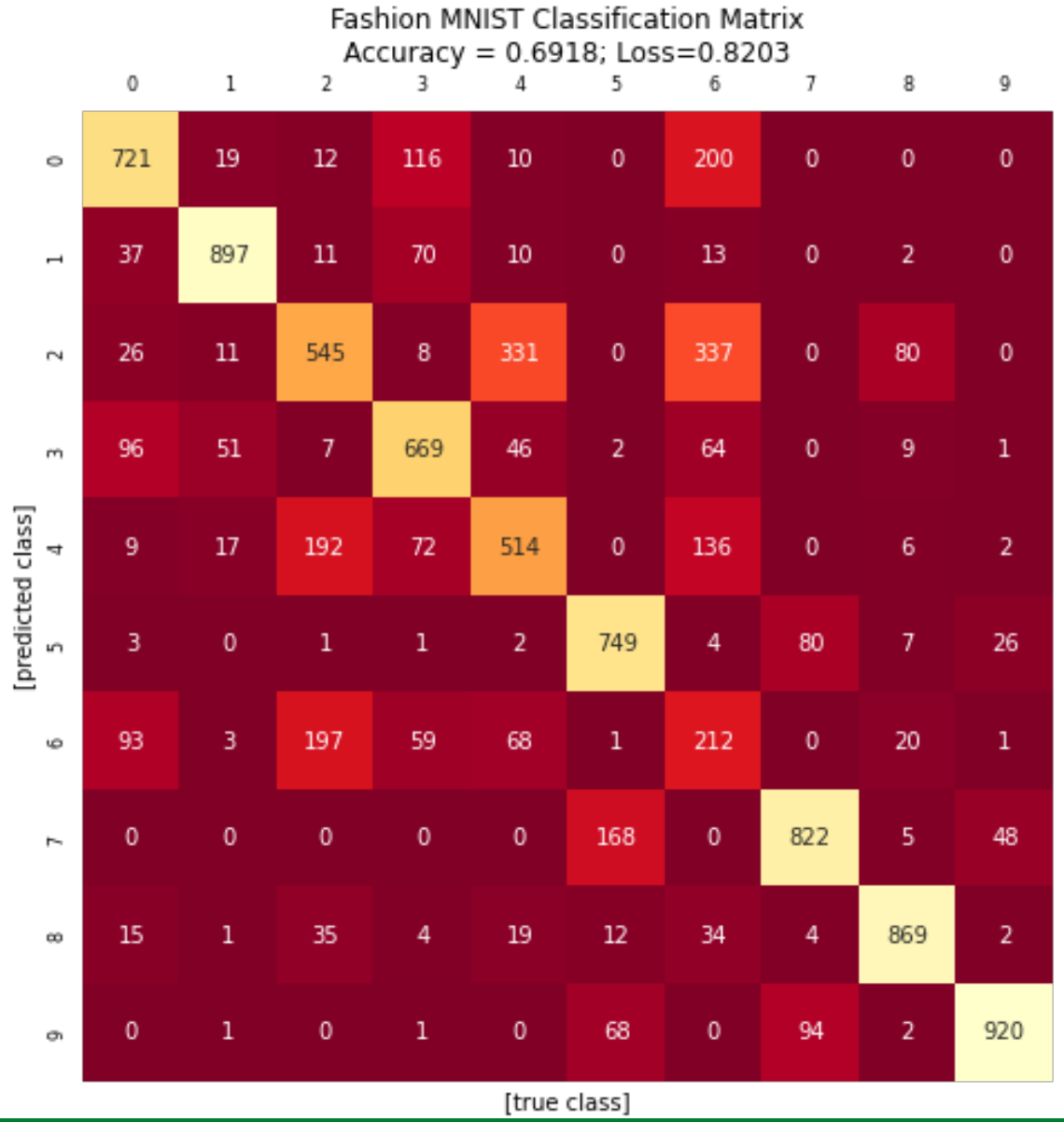
Better Model



The classification error in the baseline model decreased with each epoch until the loss reached 0.8106 in the last epoch. In the better model, the classification error also decreased with each epoch, but the loss was much less, reaching 0.2023 at the last epoch. It is unlikely that the baseline model would begin outperforming the better model if given more epochs to converge because of how far behind it is performing in terms of loss. By the 20th epoch, the better model's loss is below 0.3 but the baseline model is still above 0.8. Finally, it is unlikely either model is overfitting because the loss on the evaluation data with the baseline model is 0.8203 and with the better model the loss is 0.2500 (shown with confusion matrix on next slides).

Artificial Intelligence and Machine Learning
University of St.Gallen

Evaluation Performance – Baseline Model



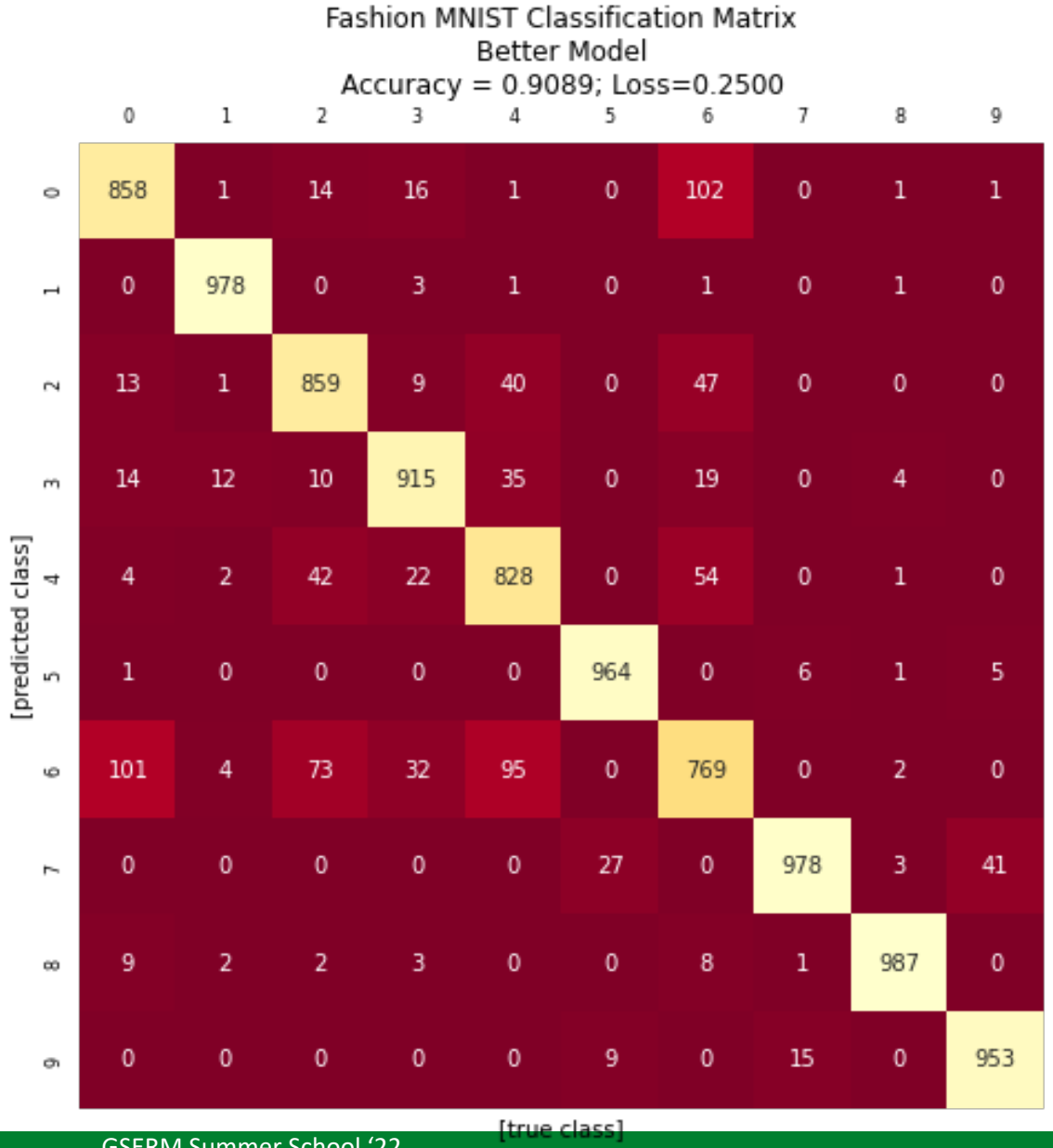
The model performance on the evaluation data had an accuracy of only 69.18%. The model had great difficulty distinguishing coat (4) from pullover (2). It incorrectly identified 331 coats as pullovers. The opposite was also true, though less so – the model incorrectly identified 192 pullovers as coats, and it also identified 197 pullovers as shirts.

The model struggled most to correctly classify shirt (6), misclassifying it as many other items – primarily pullover (2), t-shirt/top (0), and coat (4). Sandal (5) was misclassified as sneaker (7) 168 times.

The model appears to have struggled the least with bag (8), ankle boot (9), and trouser (1).

Artificial Intelligence and Machine Learning
University of St.Gallen

Evaluation Performance – Better Model



The model performance on the evaluation data achieved an accuracy of 90.89%. The baseline model struggled most to correctly classify shirt (6), and it appears that this is still the class that is most likely to be misclassified with this better model (primarily as t-shirt/top), but the performance was still much better. The baseline model classified only 212 shirts correctly, whereas this improved model was able to correctly classify 769 shirts.

Other classes where a large increase in performance can be seen are t-shirt/top (0), pullover (2), dress (3), and coat (4).

Interpretation

I believe my improved CNN is able to outperform the baseline CNN for two main reasons. The first reason relates to the architecture. The better model uses dropout regularization with a p of 0.5. This means that for the two fully connected hidden layers that have dropout regularization, the neurons within them will be randomly deactivated with a probability of 50%. The reason for dropout regularization is to stop the model from overfitting. Interestingly, I do not think that the baseline model was overfitting (possibly because I stopped the training after only 20 epochs before it converged) because the loss on the training data for the final epochs and the loss on the evaluation data were not wildly different in a way that would suggest overfitting, which would not make me immediately think that using the dropout regularization to stop overfitting was what helped my improved model perform better. However, dropout regularization, by causing the network to not rely on any particular neuron, also allows for distributing information throughout the whole network, so the improved model was able to learn and classify better for this reason, so upon further consideration, I do believe that the dropout regularization played a large role in the better performance of the improved model.

The second reason relates to the hyperparameters. Stochastic gradient descent (SGD) is used in the baseline model and the Adam optimizer is used the improved model. With gradient descent, a combination of the weight and bias is initialized randomly, and all samples are used to calculate the gradient in order to learn the direction to update the weights and biases. With SGD, which is what is used for the baseline model, one sample is used to calculate the gradient. This is more computationally efficient but computes a much noisier gradient since it loses the information contained in all the other samples. The Adam optimizer extends classic SGD by incorporating an adaptive learning rate. Interestingly, the baseline model uses a learning rate of 0.001 while the improved model uses an initial learning rate of 0.0001, so if either model was to become stuck at a local minima due to too small of a learning rate, I would have expected it to be the improved model. However, this did not happen, nor was the 0.001 learning rate paired with SGD too large of a learning rate, as we do not see the loss jumping around from epoch to epoch. I believe the Adam optimizer combined with the adaptive learning rate is another key reason that the improved model performed better.

Finally, the baseline model has 20 epochs. When plotting the loss for this model, I saw that it looked like it was starting to converge but got cut off, so in the improved model, I experimented first with 30 epochs and eventually settled on 40 epochs. This was still not enough for it to fully converge, but it did allow the model much more time to learn. However, the loss for the improved model started out lower than in the baseline model anyway, so I expect that the number of epochs is not nearly the most important reason for the improved performance. Instead, I expect that the dropout regularization and the Adam optimizer (which includes an adaptive learning rate) are the two main reasons that the improved model performed so much better.

References

- Aside from the labs and class slides, I used a few sources when deciding on the architecture and how to best visually represent my neural networks. I did not follow these references exactly – they were merely a starting point!

- Brownlee, Jason. “Gentle Introduction to the Adam Optimization Algorithm for Deep Learning.” *Machine Learning Mastery*, July 3, 2017, https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
- “Convolutional Neural Networks for Classifying Fashion-MNIST Dataset using Ignite.” Pytorch-Ignite. Viewed July 1, 2022, from https://github.com/pytorch/ignite/blob/master/examples/notebooks/FashionMNIST.ipynb
- Nutan. “PyTorch Convolutional Neural Network With MNIST Dataset.” *Medium*, May 21, 2021, https://medium.com/@nutanbhogendrasharma/pytorch-convolutional-neural-network-with-mnist-dataset-4e8a4265e118.
- Pankaj. “Fashion MNIST with Pytorch (93% Accuracy).” *Kaggle*, May 23, 2019, https://www.kaggle.com/code/pankajj/fashion-mnist-with-pytorch-93-accuracy/notebook.
- ul Hassan, Muneeb. “AlexNet – ImageNet Classification with Deep Convolutional Neural Networks.” Neurohive, October 29, 2018, https://neurohive.io/en/popular-networks/alexnet-imagenet-classification-with-deep-convolutional-neural-networks.