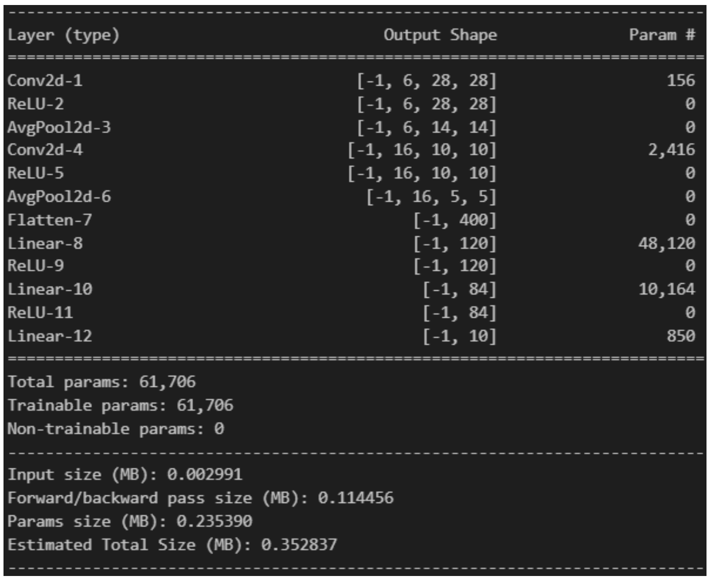
INFOMCV Assignment 4
Yannick den Boer (6803113) and Eva Kato (1179624)
(Group 15)

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.)

Base LeNet-5

- The LeNet-5 model consists of 2 sets of convolutional and pooling layers. The outputs are then flattened into a 1-dimensional tensor and passed through three linear layers.
- The convolutional layers allow us to extract local features from the image. Each convolutional kernel transforms a 5x5 area around each pixel into a feature value. By using two convolutional layers hierarchically, the second convolution can extract features that cover a bigger image area.
- The pooling layers take the average value of a 2x2 area. With the intention to downscale the parameter dimensions. This speeds up training while maintaining the most important information.
- By using a linear fully connected layer we allow flow of information between each input and output. In each neuron in a fully connected layer, features extracted by the convolutional layers are combined, learning more global features from the image. By using multiple linear layers the model can learn even more abstract or complex features.



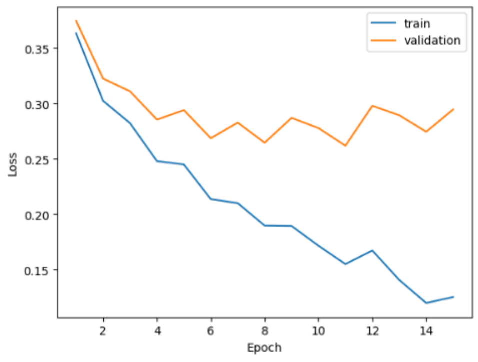
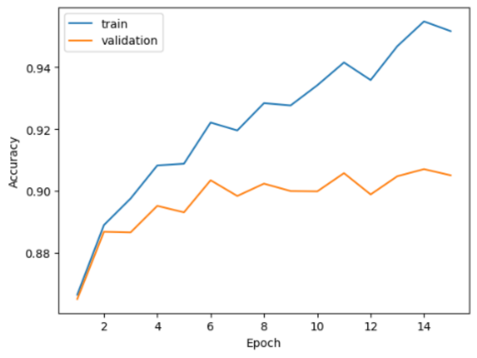
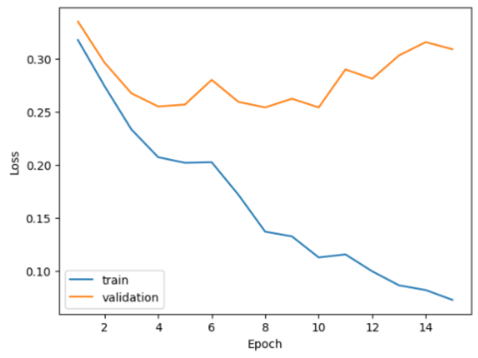
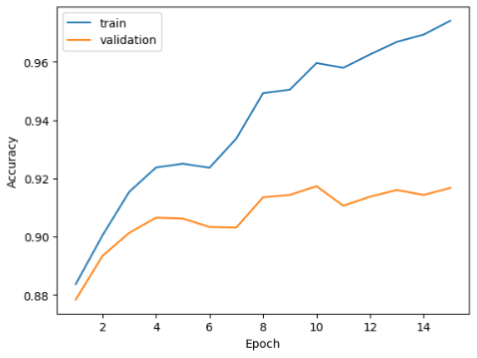
```
----------------------------------------------------------------
Layer (type)                      Output Shape         Param #
================================================================
Conv2d-1                       [-1, 6, 28, 28]             156
ReLU-2                         [-1, 6, 28, 28]               0
AvgPool2d-3                    [-1, 6, 14, 14]               0
Conv2d-4                      [-1, 16, 10, 10]           2,416
ReLU-5                        [-1, 16, 10, 10]               0
AvgPool2d-6                     [-1, 16, 5, 5]               0
Flatten-7                            [-1, 400]               0
Linear-8                             [-1, 120]          48,120
ReLU-9                               [-1, 120]               0
Linear-10                             [-1, 84]          10,164
ReLU-11                               [-1, 84]               0
Linear-12                             [-1, 10]             850
================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.002991
Forward/backward pass size (MB): 0.114456
Params size (MB): 0.235390
Estimated Total Size (MB): 0.352837
----------------------------------------------------------------
```

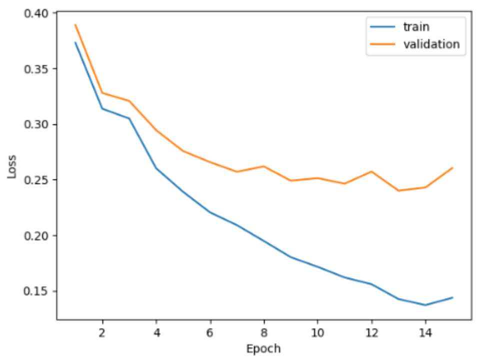
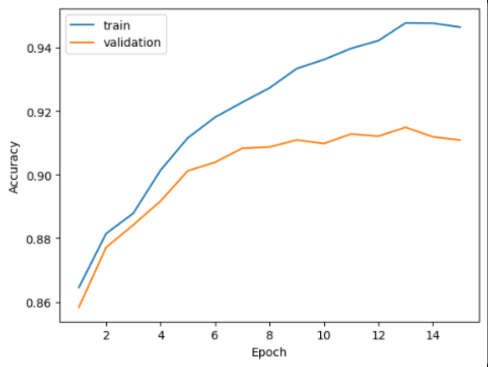
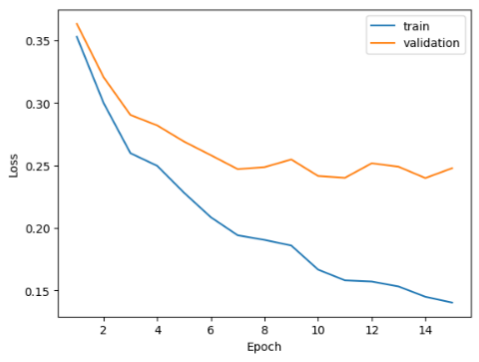
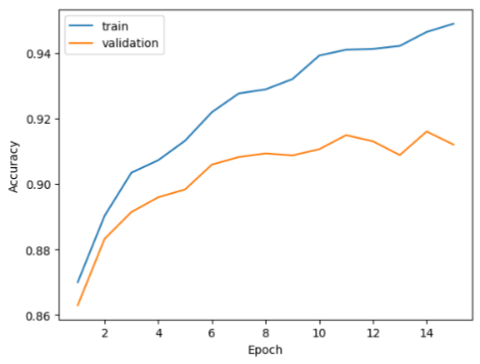
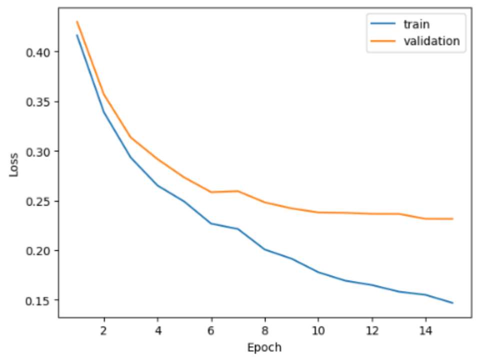
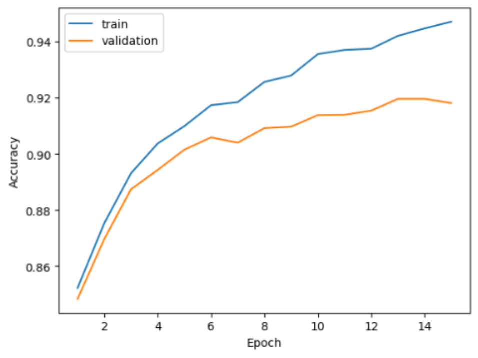
1. **Variant 1: Base + More Kernels**
   We noticed that a lot of error is due to the base model mixing up shirts, pullovers and t-shirts. In order to classify these categories better, we decided to increase the number of kernels in the convolutional layers. By using more kernels the model should be able to extract more different types of local features from the image. We increased the first layer's kernel size from 6 to 16, and the second layer's kernel size from 16 to 26.

2. **Variant 2: Variant-1 + Dropout**
   In variant 1, we noticed an increase in accuracy. However, the increase in the number of kernels allows for higher feature complexity, which allows the model to overfit more easily. In our second variant we therefore introduce a Dropout layer after the first two fully connected layers. 30% of the neurons in each layer are dropped randomly.
3. **Variant 3: Variant-2 + L2 Regularization**
   Similarly to variant 2, we aimed to find a better solution to combat our model overfitting by adding the L2 regularization term to the loss function. Our intention is to make the model learn simpler patterns by discouraging the model from using large weights, thus preventing overfitting. We add a weight decay of 0.0001.
4. **Variant 4: Variant-3 + Decreased Learning Rate**
   We noticed the previous two variants caused oscillation in the loss, so we decided to half the learning rate (0.001 ⟶ 0.0005). As a result, the optimizer takes smaller steps and potentially overcome local minima, thus converging to a better solution.

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)

	Loss graph	Accuracy graph
Base LeNet-5		
Variant 1		

Variant 2		
Variant 3		
Variant 4		

Link to your model weights

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

https://drive.google.com/drive/folders/1ASh0s9kWP7L4qjzG09LLKMZ8KAeRiw1d?usp=sharing

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 (%)	Epoch
Base LeNet-5	95.5%	90.7%	14
Variant 1	95.9%	91.7%	10
Variant 2	94.8%	91.5%	13
Variant 3	94.8%	91.6%	14
Variant 4	94.5%	92.0%	14

Discuss your results in terms of your model

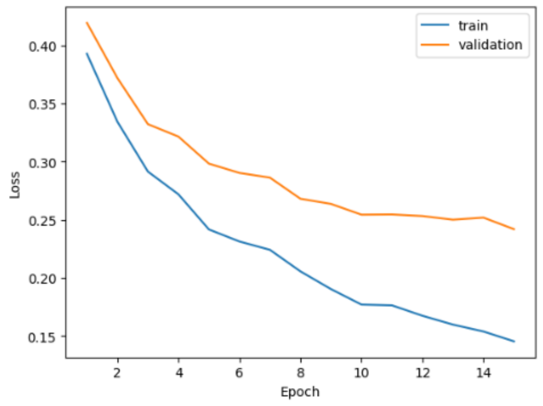
(Discuss the results in terms of complexity, type of layers, overfitting measures, etc. Make pair-wise comparisons between the four variants and the baseline model. Approx. 0.5 page.)

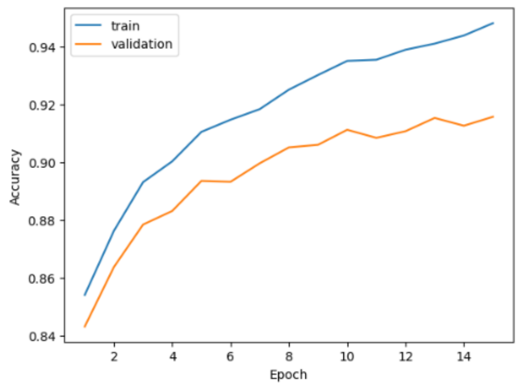
1. **Base LeNet-5 vs Variant 1**
   We see an increase in top-1 validation accuracy. Which is caused by the increased complexity that was introduced by adding more kernels. However, due to this complexity increase the model starts overfitting more quickly: after the tenth epoch the loss only gets worse.
2. **Variant 1 vs Variant 2**
   The dropout layers prevent co-adaptation between the neurons. Making the model more generalised and preventing overfitting. We can see the regularisation works because the loss is no longer increasing drastically after a certain number of epochs. Another consequence of regularisation is that the training accuracy is lower. But the validation accuracy and the training accuracy are closer together.
3. **Variant 2 vs Variant 3**
   We see no significant difference between variant 2 and variant 3. Since experimenting with higher values for weight_decay lead to worse performing models, we conclude that there is not much more to gain from regularisation.
4. **Variant 3 vs Variant 4**
   The loss graph of variant 4 is smoother as expected due to the lower learning rate, which also allows it to get a slightly higher accuracy (although the change is very small). For the same reason, variant 4 will also be more susceptible to getting stuck in local minima.

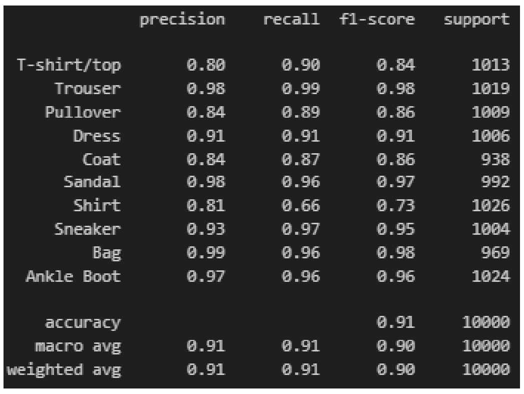
Discuss the differences between the two models evaluated on the test set
(Discuss the potential causes for (lack of) differences in terms of architecture. Also compare the results of each model to the training performance. Approx. 0.5 page.)

We use **Variant 4** to compare the two trained models (train vs. train+val). The model trained on solely the training data performed very slightly better, with a highest accuracy of 91.6% vs. 91.3% on the train+validation. We can reason that since the model architecture includes a dropout layer and L2 regularization, the model generalizes well even on the smaller training set, less relying on specific features and patterns and preventing overfitting. Furthermore, it is possible that the architecture of our model is already sufficiently complex to capture the patterns in the data, so the extra benefit of training on combined dataset may be limited.

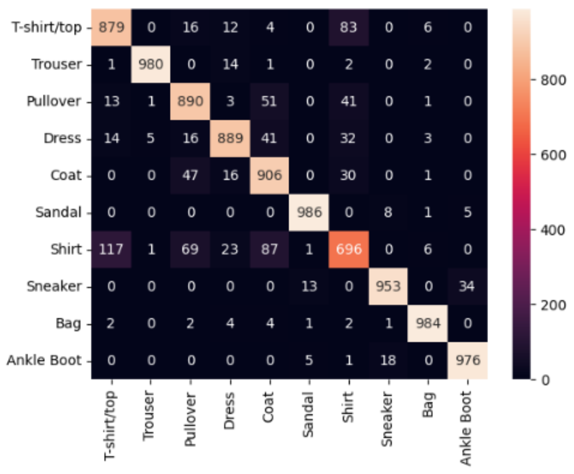
- **Train**
  top1 accuracy: 94.8, 91.6 (epoch 15)





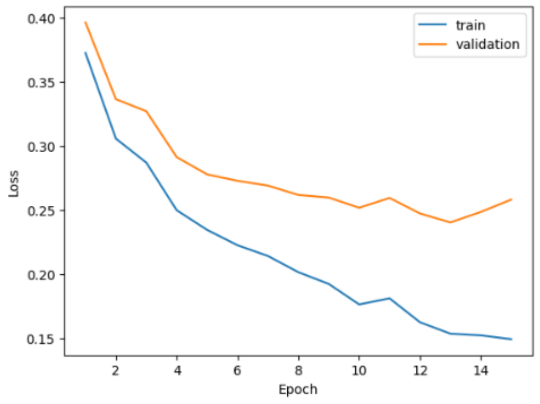


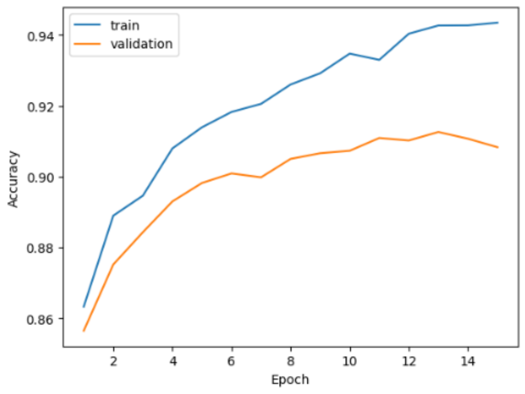
```
              precision    recall  f1-score   support

 T-shirt/top       0.80      0.90      0.84      1013
     Trouser       0.98      0.99      0.98      1019
    Pullover       0.84      0.89      0.86      1009
       Dress       0.91      0.91      0.91      1006
        Coat       0.84      0.87      0.86       938
      Sandal       0.98      0.96      0.97       992
       Shirt       0.81      0.66      0.73      1026
     Sneaker       0.93      0.97      0.95      1004
         Bag       0.99      0.96      0.98       969
  Ankle Boot       0.97      0.96      0.96      1024

    accuracy                           0.91     10000
   macro avg       0.91      0.91      0.90     10000
weighted avg       0.91      0.91      0.90     10000
```

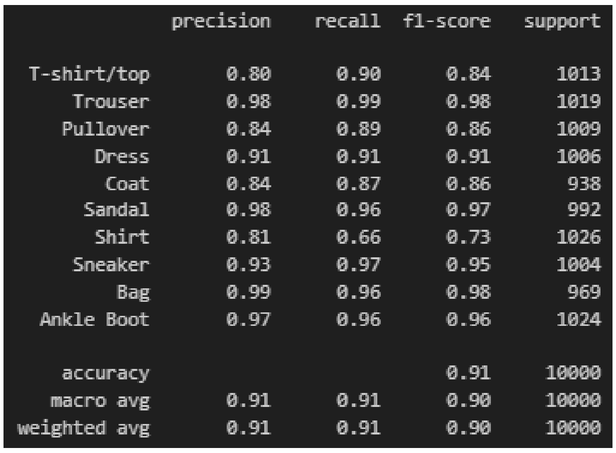


	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle Boot
T-shirt/top	879	0	16	12	4	0	83	0	6	0
Trouser	1	980	0	14	1	0	2	0	2	0
Pullover	13	1	890	3	51	0	41	0	1	0
Dress	14	5	16	889	41	0	32	0	3	0
Coat	0	0	47	16	906	0	30	0	1	0
Sandal	0	0	0	0	0	986	0	8	1	5
Shirt	117	1	69	23	87	1	696	0	6	0
Sneaker	0	0	0	0	0	13	0	953	0	34
Bag	2	0	2	4	4	1	2	1	984	0
Ankle Boot	0	0	0	0	0	5	1	18	0	976

- **Train+Val**
  top1 accuracy: 94.3, 91.3 (epoch 13)







```
             precision    recall  f1-score   support

 T-shirt/top       0.80      0.90      0.84      1013
     Trouser       0.98      0.99      0.98      1019
    Pullover       0.84      0.89      0.86      1009
       Dress       0.91      0.91      0.91      1006
        Coat       0.84      0.87      0.86       938
      Sandal       0.98      0.96      0.97       992
       Shirt       0.81      0.66      0.73      1026
     Sneaker       0.93      0.97      0.95      1004
         Bag       0.99      0.96      0.98       969
  Ankle Boot       0.97      0.96      0.96      1024

    accuracy                           0.91     10000
   macro avg       0.91      0.91      0.90     10000
weighted avg       0.91      0.91      0.90     10000
```

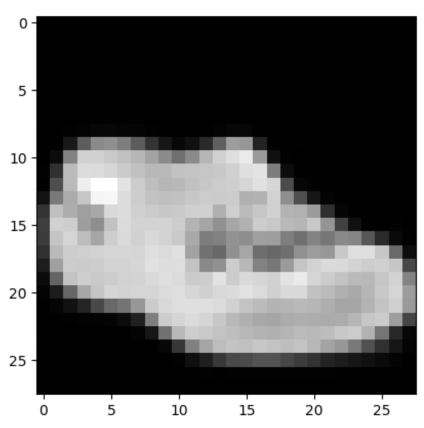
Choice tasks

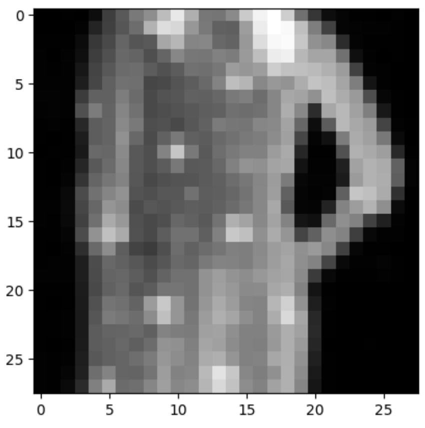
(Indicate which ones you did, and how you did them; Approx. half a page.)

<u>Using the Fashion Product Images (Small) dataset</u>

First, we filter the new dataset because there are some irrelevant images that don't appear in the MNIST, like fragrances and accessories. Then, we map the classes from the new dataset to the classes of MNIST dataset. For example, the new dataset differentiates "jeans" and "track pants" which should be labelled as "Trouser" in the MNIST set. We make these labels to correspond to the same order as the MNIST (labelled 0-9). We also noted that none of the images in the new dataset correspond to "Ankle Boot" in MNIST, so that was completely omitted. Then the images are preprocessed to fit the images in the MNIST set by center cropping, resizing to 28x28 pixels, grayscaling, and then inverting the colors. The colors are inverted because the MNIST set uses black backgrounds while the new dataset uses white backgrounds.

These are example preprocessed images, labelled as "sneaker" and "T-shirt/top" respectively:





We used Variant-1 to test the new dataset. The best accuracy was 34.65% (epoch 13). Based on the precision-recall table below, we can see the class imbalance on the new dataset, where classes like "T-shirt/top" and "Trouser" are supported well while there are no correct classifications for "Pullover", "Dress", "Coat", "Shirt", and "Ankle Boot". This is expected because so many of the upper body clothing items images are shown being worn by a person, which makes it a lot trickier to classify amongst each other, especially when converting to grayscale.

```
              precision    recall  f1-score   support

 T-shirt/top       0.91      0.44      0.59     18714
     Trouser       0.71      0.69      0.70      3324
    Pullover       0.00      0.00      0.00         0
```

```
       Dress       0.00      0.00      0.00         0
        Coat       0.00      0.00      0.00         0
      Sandal       0.08      0.03      0.04      2355
       Shirt       0.00      0.00      0.00         0
     Sneaker       0.61      0.02      0.04      8844
         Bag       0.17      0.55      0.26      3674
  Ankle Boot       0.00      0.00      0.00         0

    accuracy                           0.35     36911
   macro avg       0.25      0.17      0.16     36911
weighted avg       0.69      0.35      0.40     36911
```