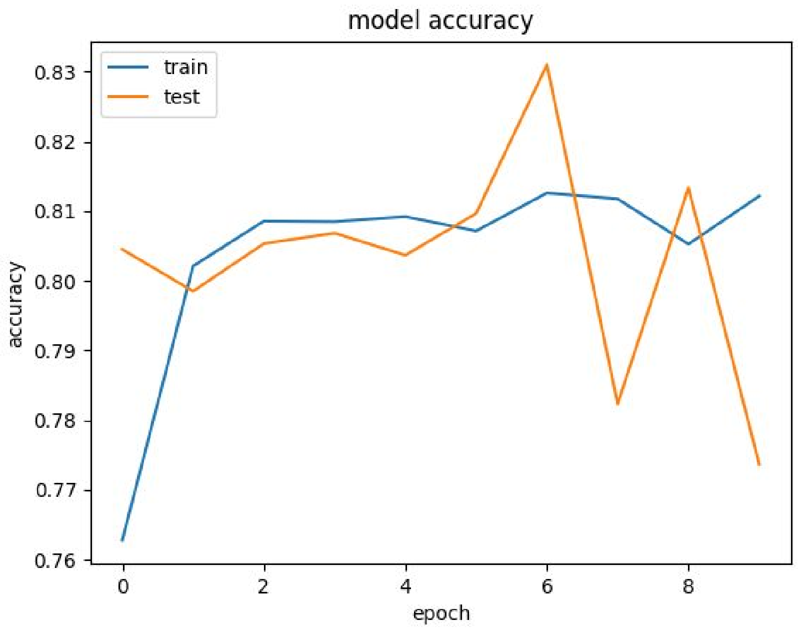
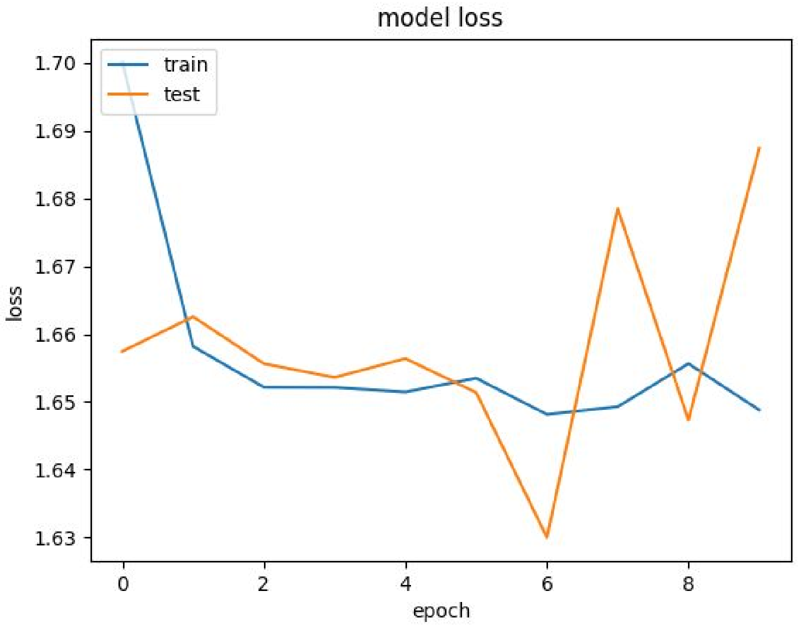
MLP

```
Model: "sequential"
_________________________________________________________________
Layer (type)              Output Shape            Param #
=================================================================
dense (Dense)               (None, 128)             100480
_________________________________________________________________
dense_1 (Dense)             (None, 128)              16512
_________________________________________________________________
dense_2 (Dense)             (None, 10)              1290
_________________________________________________________________
activation (Activation)     (None, 10)             0
=================================================================
Total params: 118,282
Trainable params: 118,282
Non-trainable params: 0
_________________________________________________________________
```

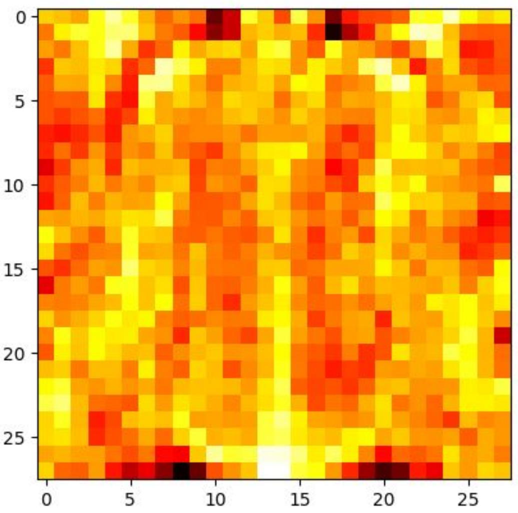
I used a two layer MLP with softmax activation and no regularization methods. It uses the Keras backend and can achieve just over 80% accuracy.

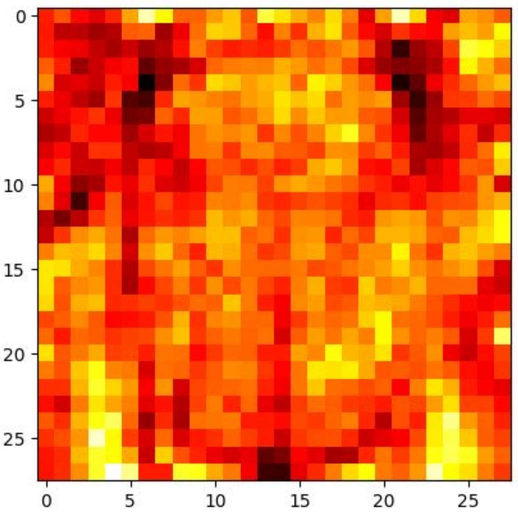
Why isn't MLE a good method for loss here?
The output space is discrete, so a notion of "distance to the correct answer" is nonsensical.





As one may observe from the preceding graphs, the accuracy does not seem to full converge and the model may benefit from additional epochs. In the future, this will be an improvement that I will consider.



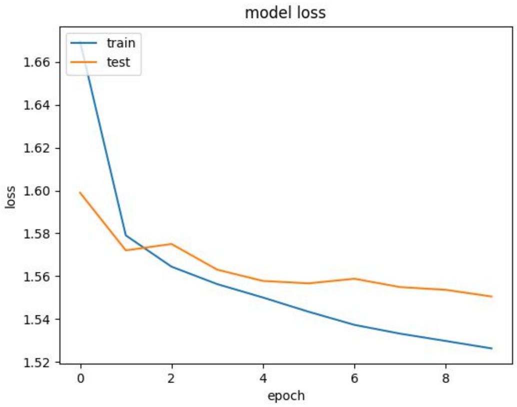


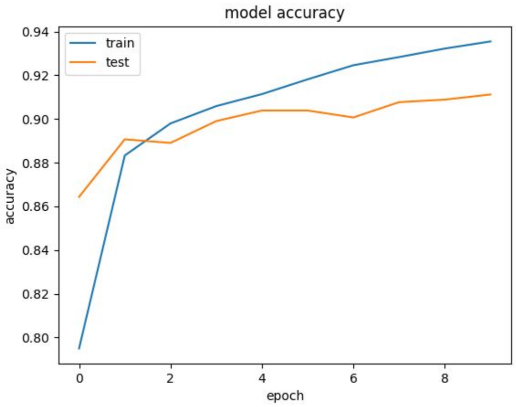
These don’t really resemble specific articles of clothing, but you can clearly see some symmetries in the weightings.

CNN

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 64)        640
_________________________________________________________________
activation (Activation)      (None, 28, 28, 64)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 12544)             0
_________________________________________________________________
dense (Dense)                (None, 128)               1605760
_________________________________________________________________
activation_1 (Activation)    (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
_________________________________________________________________
activation_2 (Activation)    (None, 10)                0
=================================================================
Total params: 1,607,690
Trainable params: 1,607,690
Non-trainable params: 0
_________________________________________________________________
```

As you can see from the output above, my model uses a single convolution and pooling, followed by a flattening layer and two dense layers, with each step having an activation between.

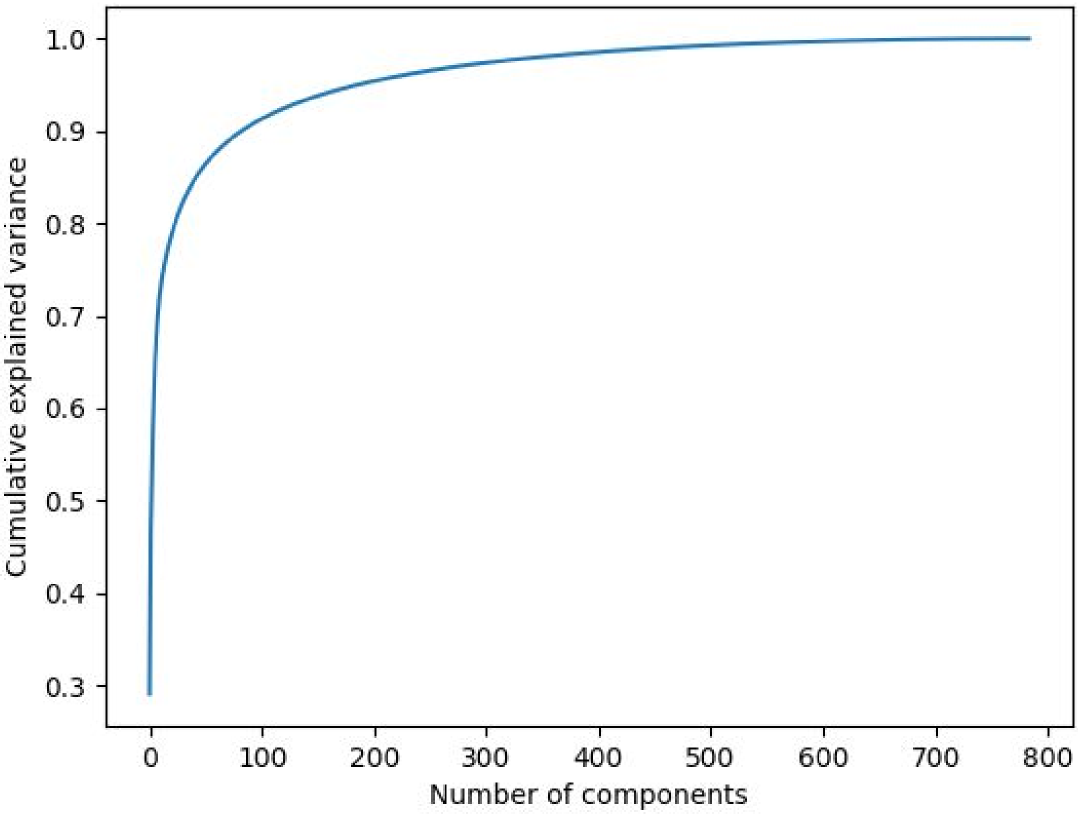




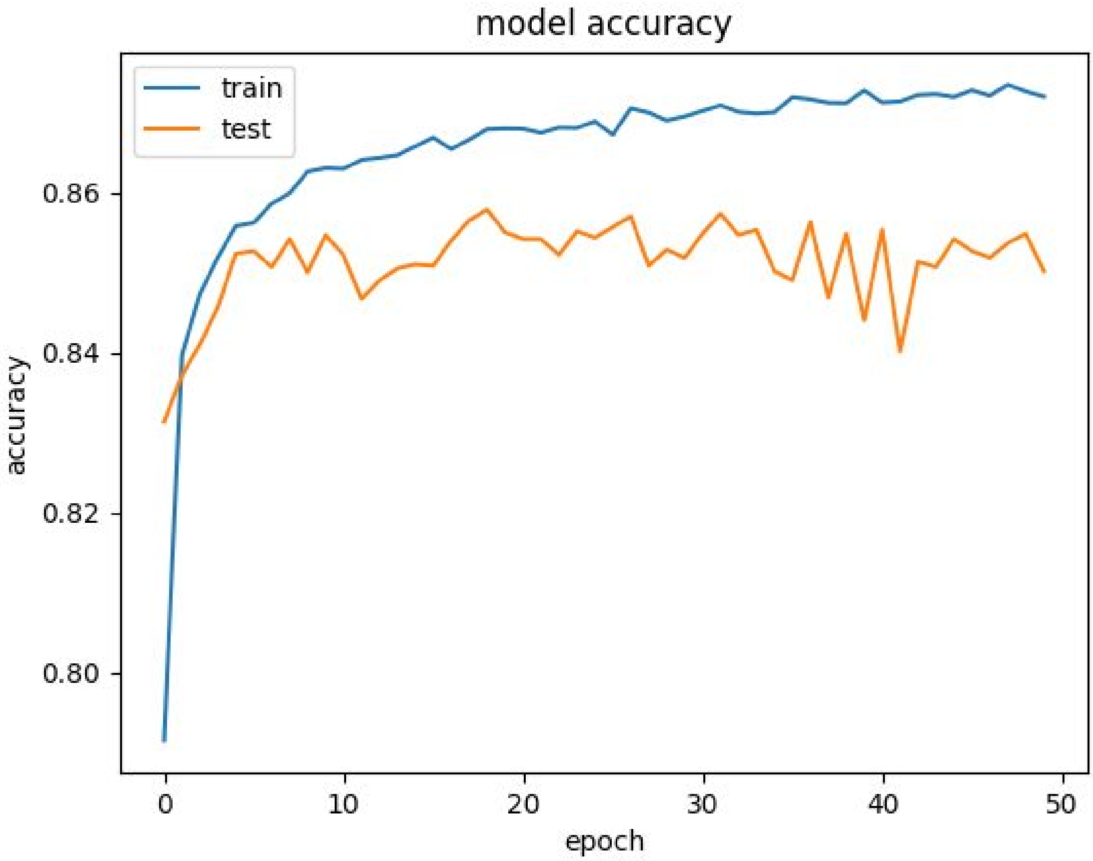
As you can see from the above graphs, the model may benefit from additional epochs. I have therefore increased the number of epochs to 50 for the grading mode.

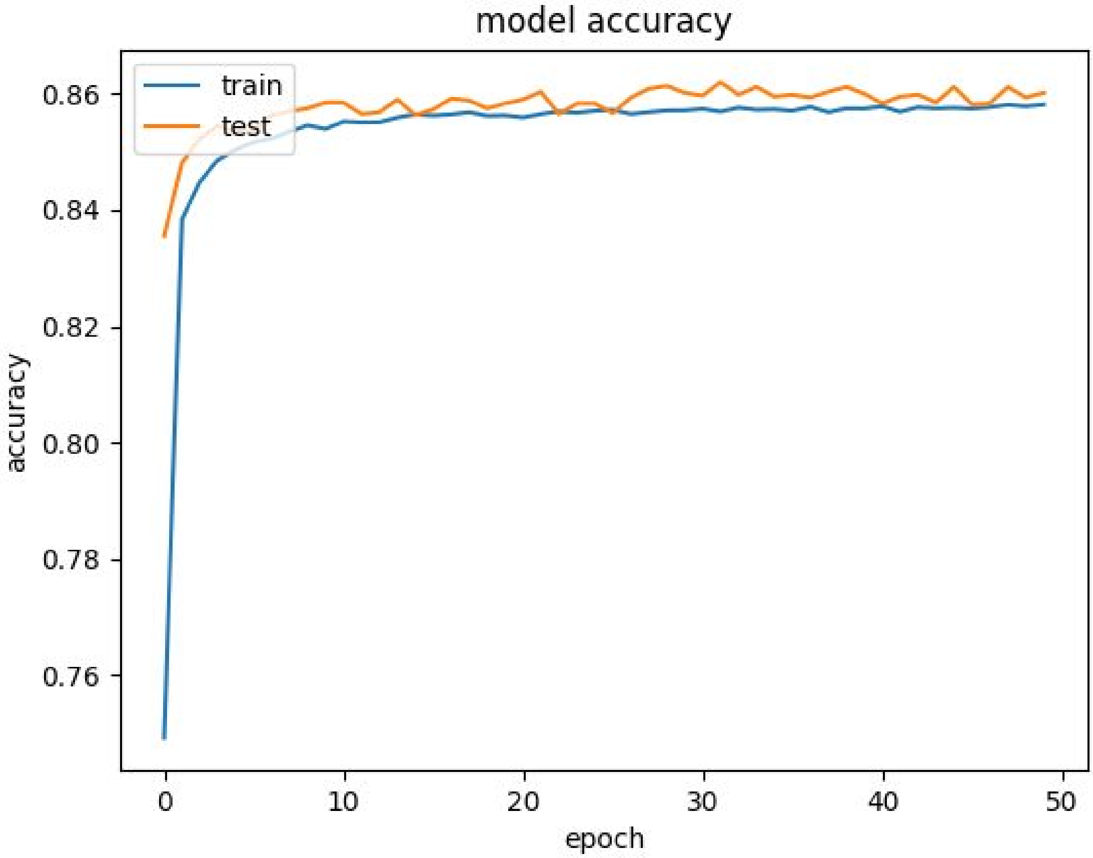
How many matrices?
1
What are the dimensions?
28x28
After maxpooling?
27x27

PCA



I chose a component value of 350, because it looks like that explains most of the variance (the graph above should suffice for showing the error at each level, since explained variance and error are directly related). As you can see from the two graphs below, the PCA seems to help with convergence slightly (the second is with PCA):


model accuracy
train
test
accuracy
0.86
0.84
0.82
0.80
epoch
0
10
20
30
40
50


model accuracy
train
test
0.86
0.84
0.82
0.80
0.78
0.76
accuracy
0
10
20
30
40
50
epoch