INFOMCV

Pepe Cano-Manuel Claver and Ana San Román

1)_Description and motivation of your baseline model and four variants

a. Baseline Model

The baseline model utilizes the Annex 1 architecture. It starts with a convolution layer with 32 filters and a kernel size of 3x3. Due to the small sizes of the input images, this kernel size was chosen to obtain more complex characteristics (better than those using a kernel of 2x2) while still being small enough for the size of the input images of 28x28. Because of the low image resolution, the kernel size is kept constant in all the convolutional neural networks that follow.

Meanwhile, the number of filters is increasing from 32 to 64. This is because the filters learn more abstract and complex features as we advance through the network, which are combinations of the simpler features learned in the earlier layers. In order to recognize increasingly complicated patterns in the input data, the network can learn more varied and complex features by adding additional filters to each succeeding convolutional layer. Another factor is that the networks' input layer, which gets the raw data, is always noisy. On the other hand, larger number of filters with smaller kernel were chosen instead of bigger kernels with smaller filters. This is because, although the receptive field is kept constant, we are more flexible.

Then, after each CNN pooling is used to improve the performance helping the network to generalize better and reduces overfitting. Particularly, *maxpooling* reduces translation invariance (allows the network to learn features invariant to small shifts in the input), the reduced computational cost (reduces the number of parameters) and improves generalization performance (by selecting the strongest activations help remove some of the noise and irrelevant features in the input).

Then, after each convolutional layer and *maxpooling* layer a dropout of 0.25 is assessed. It is used as a regularization technique, forcing the network to be redundant, eliminating co-adaption, and preventing overfitting. Basically, forcing some neurons to be discriminative.

Finally, the architecture of Conv-MaxPooling-Dropout was repeated so that the spatial dimensions is reduced while extracting sets of different features. By repeating the network, it can learn a hierarchy of increasingly complex features, with each layer building upon the representations learned in the previous layer.

Finally, there is a flatten layer to get a vector and input it into the dense layer with size 128. Then, another dropout is used to keep preventing regularization. And the final dense connective layer of size 10, because of the 10 different classes.

As of the activation maps, ReLU was chosen as the activation function, since it avoids the vanishing gradient problem and it is non-linear, allowing the model to learn more complex patterns. Then, the softmax function is used in the last dense layer. This is because the Softmax function converts the output of the last layer into a probability distribution over the different classes. It is especially useful in multiclass classification.

Moreover, as a learning rate Adaptive Moment Estimation (Adam) algorithm is used. It helps achieve a faster convergence, since the optimizer can adjust the learning rate based on the history of the gradients and the magnitude of the weight updates, avoiding getting stuck in a local minima.

b. Variant 1: modification of kernel size

Increasing the kernel size has many effects on the performance of a CNN model. It increases the receptive field, allowing them to capture more information from the input images. Also, while a small kernel size might only capture basic features such as edges, larger ones can capture more complex features like curves or circles. Moreover, increases robustness to noise and might help reducing overfitting. Because of that, a kernel size was increased to 7 in the first Convolutional Neural Layer.

c. Variant 2: batch normalization

For now, the regularization tool used in the baseline model was dropout. Dropout can be thought of as a form of ensemble learning, where multiple sub-models are trained and combined during prediction. The effect of dropout is to reduce the variance of the model, which can help to prevent overfitting. On the other hand, the effect of batch normalization is to reduce the internal covariate shift in the model, which can help to improve its generalization performance. For that, in this variant model batch normalization was added after the convolutional layer, keeping the dropouts after *maxpooling*. In this sense, more regularization is applied on the model with the aim of preventing even more overfitting.

d. Variant 3: learning rate

In comparison to the adaptative-like Adam learning rate algorithm used in the baseline model, a study with a fixed learning rate is assessed. Momentum weight considers changes in previous weights when updating current weights. In high curvatures it maintains a stable learning rate while speed up the learning in directions of low curvature. For that, the learning rate is set to 0.01 and the momentum to 0.9. A learning rate of 0.01 means that the weight updates will be proportional to 0.01 times the gradient of the loss function with respect to the weights. A momentum of 0.9 means that the weight updates will be a weighted average of the current gradient and 90% of the previous iteration's gradient update. This combination of hyperparameters can help to balance the tradeoff between convergence speed and stability.

e. Variant 4: sigmoid activation function

While ReLU focuses on preventing vanishing gradients, Sigmoid activation functions help on not blowing up activations. For that, the last variant corresponds to the use of sigmoid as the activation function on the Conv2D and the Dense layer, except for the last one where softmax is kept.**Link to model weights**

Model name	**Training top-1 acc. (%)**	**Validation top-1 acc. (%)**
Baseline model *(model)*	0.9188	0.9173
Kernel size model *(model_kernel)*	0.9017	0.9007
BN model *(model_norm)*	0.9180	0.9191
Learning Rate *(model_LR)*	0.9027	0.9067
Sigmoid model *(model_sigmoid)*	0.8673	0.8765

Table 1. Training and validation acc. for 5 CNN variants.

2) Discussion in terms of models

As seen in Table 1, all five models achieve a good performance on the validation set. The baseline model has the highest accuracy in training and the second in validation, thus, the chosen layers are already making a good combination.

Firstly, the baseline model differs from the first variant model in the first and second convolutional layers, since the kernel size has been augmented from 3x3 to 7x7. The resulted fall in accuracy can be due to a loss in spatial information since more neighbor pixels are being consider which can make the distinguishing of same structures harder for the network. Moreover, a higher kernel can lead to missing small but important features that are important for the classification, thus, this can also be contributing to the decrease. It could also be caused by an overfitting factor produced from the more parameters the networks have to train.

On the other hand, the second variant performs better than the baseline model in validation. The three batch normalization layers added after each convolutional layer and the first dense layer before the activation function seems to be regularizing and preventing overfitting. Moreover, it is helping the network to learn and improving the performance as a consequence of normalizing the input to the activations in each layer maintaining the nonlinearity.

Furthermore, the third variation uses a stable learning rate throughout the training with a momentum factor. Although the momentum helps to accelerate the convergence of the

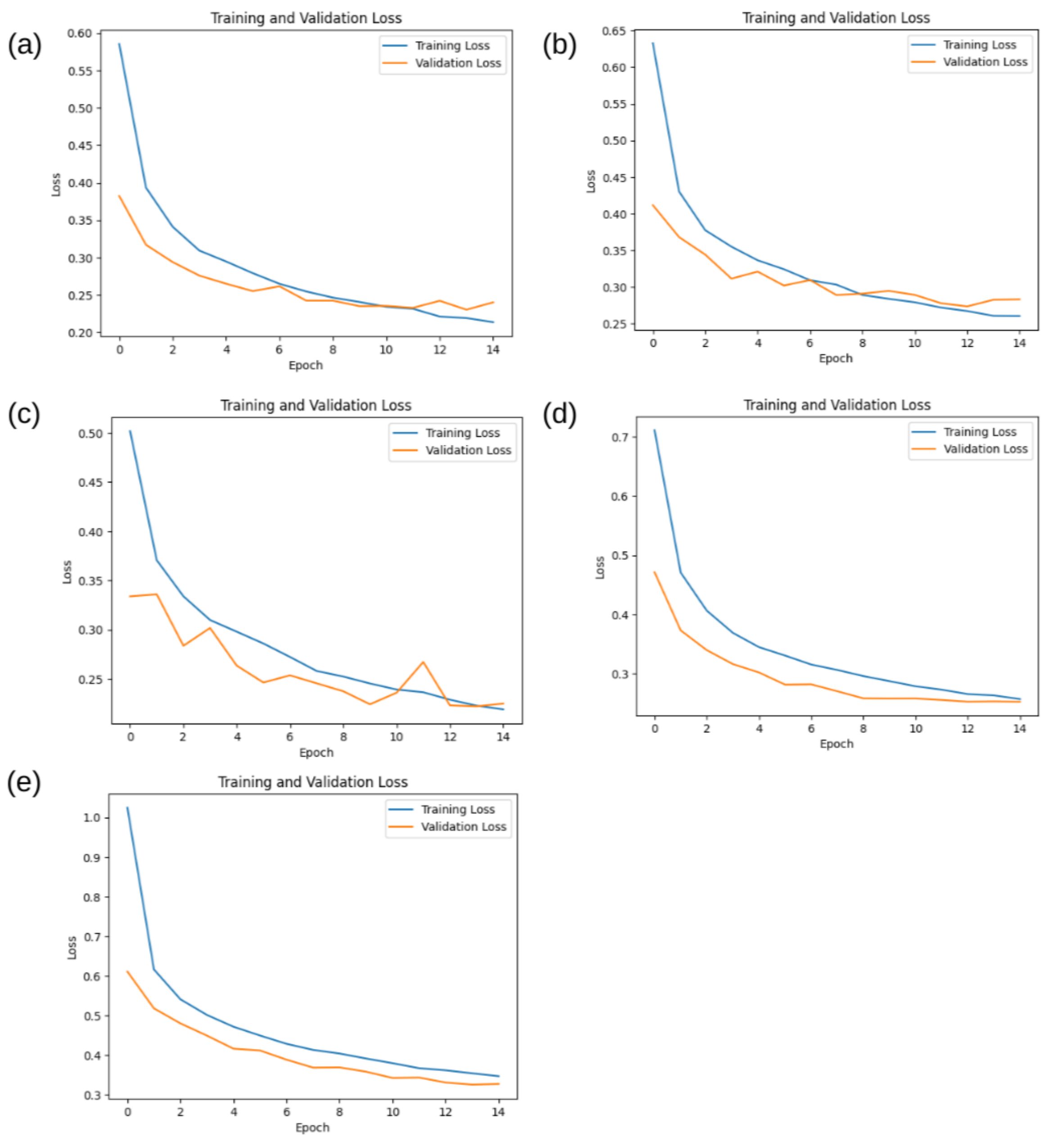


Figure 1. Training and validation loss curves for: (a) baseline model, (b) kernel size model, (c) batch normalization model, (d) lear_ning rate model and (e) activation function model.

algorithm towards the global minimum of the loss function by incorporating information about the previous updates to the weights into the current update, the Adam optimizer used in the baseline model seems to perform better. This can be due to its adaptative learning rate which tunes the momentum and learning rate automatically for each parameter based on its own behavior, therefore, is taking into account more training information and not using the same values on all iterations.

Finally, the last variation from the baseline model uses the sigmoid function as the activation function instead of ReLu. Out of all the variants, this is the one that performs worse. The reason for getting worse results might be because the sigmoid does not avoid the vanishing gradient problem since it has a limited range of output values that cause gradients to vanish during backpropagation in deeper layers. On the other hand, ReLu's unbounded output range helps avoiding this problem. Another reason could be related to its actual functionality, given that sigmoid is more suited for producing probabilities in the output layer and ReLu is typically preferred in the hidden layers, thus, implementing sigmoid in the hidden layer might be slowing down the training and not representing the full range of possible values.

As seen in Figure 1, all models perform relatively well in relation to the validation and training loss. However, it is remarkable to notice that the baseline model (a), the kernel model (b), and the BN model (c) reach some kind of overfitting, since the validation curve surpasses the training loss curve. Specially, for the baseline model. Also, it is n oticeable that the BN model (b) reduces overfitting with respect to the baseline one, due to the use of a more regularization. For the Sigmoid (e) model overfitting is never reached, but the loss is considerably higher than that of the original baseline (a). However, both lines fit relatively well in all the figures, thus can be concluded that overfitting is not a major issue in none of the models.

3) Discuss the differences between the two models evaluated on the test set.

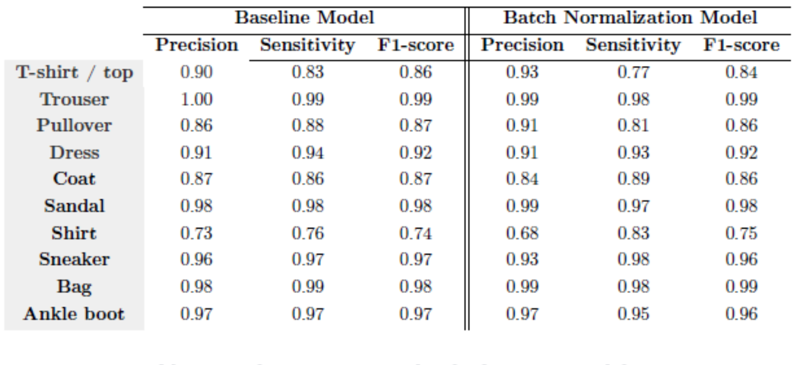
	Baseline	**BN model**
Accuracy	0.9156	0.9084

Table 2. Test acc. for the best two models

In relation to the accuracy, both models perform similarly but the baseline one has a higher overall value (Table 2). This might mean that the regularization is too strong and leads to underfitting, or that the training process is more difficult due to the introduction of randomness in the model.

In terms of precision (Table 3), the shirt class is the one that performs the worst on both models. Since the change of the second model is on regularization, it is expected that it improves the generalization performance of a model by reducing overfitting to the training data. That happens for classes like Pullover or Bag. However, if the regularization technique reduces the model's ability to learn complex class-specific features, it may lead to lower precision on some classes that require these features to be correctly classified. That happens to classes like Shirt or Coat where the baseline model performs better.

Continuing in Table 3, for the sensitivity, the Shirt is the worst class in the Baseline Model while T-shirt class performs worst for the BN model. On the other hand, Sandal, Trouser, Sneaker, Bag, and Ankle Foot are the best performing classes on both models. This means that both models have a low rate of false



	Baseline Model			Batch Normalization Model		
	Precision	Sensitivity	F1-score	Precision	Sensitivity	F1-score
T-shirt / top	0.90	0.83	0.86	0.93	0.77	0.84
Trouser	1.00	0.99	0.99	0.99	0.98	0.99
Pullover	0.86	0.88	0.87	0.91	0.81	0.86
Dress	0.91	0.94	0.92	0.91	0.93	0.92
Coat	0.87	0.86	0.87	0.84	0.89	0.86
Sandal	0.98	0.98	0.98	0.99	0.97	0.98
Shirt	0.73	0.76	0.74	0.68	0.83	0.75
Sneaker	0.96	0.97	0.97	0.93	0.98	0.96
Bag	0.98	0.99	0.98	0.99	0.98	0.99
Ankle boot	0.97	0.97	0.97	0.97	0.95	0.96

Table 3. Evaluation metrics for the best two models.

negatives for those classes. It is remarkable the discrepancy in sensitivity in Pullover.

In terms of the F1-score (Table 3), the best performing class on both models is Bag. It means that the model has a high proportion of true false negatives for that class. On the other hand, the worst performing class is Shirt on both.

In conclusion, the best performing class on both models is Bag, while the worst one is Shirt. This might indicate that the regularization is not a good solution for a good Shirt class estimation, nor for the improvement of any of the other classes. In general, the lower performing classes and the better performing ones are maintained on both models. But the accuracy indicates that the Baseline model is slightly better on the classification task.

In comparison to the training performance, both models perform worse in the test than both validation and training top-1 accuracy (Table 1). However, the difference is insignificant, so can be concluded that the model is not "memorizing" the training data but rather learning general patterns. Moreover, the validation top-1 accuracy on both is similar, which is an expected phenomenon since the data introduced on the training and validation sets are similar. It is expected that the training accuracy is higher than the other ones since the metric is calculated on the same images that are used for training the model.

4) Choice tasks

In this assignment, choice tasks number 1, 2, 3 and 5 were implemented. Out of the two best performing models, the baseline model was used for these tasks.

First, the learning rate scheduler was set on the baseline model. And the data augmentation too, separately. As out of the two techniques only the LR scheduler improves the performance, this was also used on the K-folds algorithm.

For choice task 2, a function called *step_decay*, which decreases the learning rate at a 1/2 of its value every 5 epochs, was implemented. The initial learning rate was set to 0.001. To add this function into the training of the model the *LearningRateScheduler* function from *keras* was used. This function allows you to update the learning rate value on the optimizer following the requirements of the function provided, in this case *step_decay*.

Then, for data augmentation the transformations chosen were rotation, flipping to the rotated image and addition of gaussian noise in each of the images from the training data set. The gaussian noise follows a probability distribution of mean 0 and variance of 0.01. An example of the data augmentation applied for one image is shown in Figure 2. The data was only transformed on the previously divided training set, not on the validation set.

Thirdly, k-fold cross validation was performed splitting the training data into 5 folds and then looping over the folds. For each fold, the data is split into training and validation sets, 4 folds for training and 1 for validation. Then, the model is compiled and trained using the training and validation sets for this fold. Moreover, the learning rate scheduler improvement to adjust the learning rate during training is taken into account. Finally, the confusion matrix was obtained for the best performance model out of all, which was the baseline model using k-fold and the learning rate tasks. The results can be seen in Table 4 and Table 5.

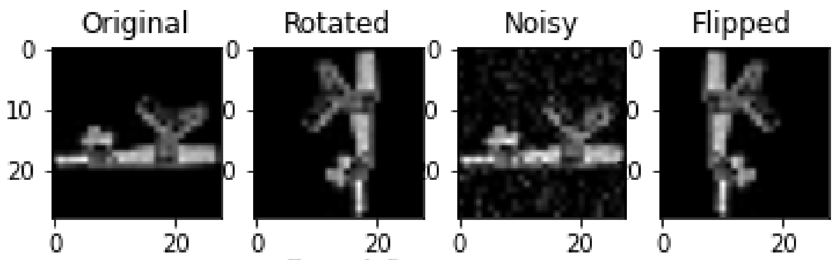


Figure 2. Data augmentation

Metric	Best model (baseline)	Choice 2: LR	Choice 5: Data aug.
Test Accuracy	0.9156	0.9180	0.8950

Table 4. Model comparison on test set for the best model and choice tasks 2 and 5

Model name	Train. top-1 acc. (%)	Val. top-1 acc. (%)	Test accuracy
Best model *(baseline)*	0.9188	0.9172	0.9156
3: k-fold & LR & best model	0.9548	0.9708	0.9259

Table 3. Comparison between best model and choice task 3

As seen in Table 4 adding a learning rate which drops half of its value every 5 epochs raised slightly the performance of the baseline model. The reason for this could be because is helping the training to converge faster and obtaining a best accuracy at the same time. On the other hand, data augmentation reduced the model performance in the test set. This might be due to a wrong choice of transformations, adding redundancy to the model. This could make sense since is a clothing dataset. For example, flipping a sandal vertically does not make sense since its orientation is important, where a shoe is not seen vertically.

On the other hand, Table 5 shows a comparison between the baseline model and the improved training model with both k-fold and learning rate tasks for training, validation and test. It is clear that making use of the k-folds methodology the model is optimizing the learning process. A reason for this might be due to the fact that is not relying on a single train-validation split that may not be representative enough of the entire dataset. This also relates to overfitting, since is not using the same training data each time, therefore is helping to generalize better and finally to assist a better hyperparameter tunning.

Lastly, the confusion matrix (Figure 3) was obtained for the best performance of all the models, which was the baseline model using k-fold and the ½ learning rate drop in training. What pops out the most in the confusion matrix is the model misclassification for the shirts into T-shirts/tops (94 FP), pull-overs (52 FP), dresses (23 FP) and coats (41 FP). It makes sense the model confuses a shirt with similar clothing items such as those and not with, for example, a shoe. Nevertheless, it still is the class with more misclassifications of all, thus, the model might have a problem in learning the features from that class. Overall, shirts and t-shirts are the clothing items more mistaken. However, the model tends to predict more a t-shirt when it is truly a shirt (94) than predicting a shirt when is truly a t-shirt (79). Moreover, the model tends to predict pull-overs when a coat or a shirt are the true label, which also makes sense for being similar in some way.

On the other hand, classes such as trousers, any kind of shoes and bags have a good ratio of classification. Certainly, the model has a slight misclassification between sandals and sneakers or sneakers with ankle boots, but it is a reasonable amount considering they are from the same clothing department.

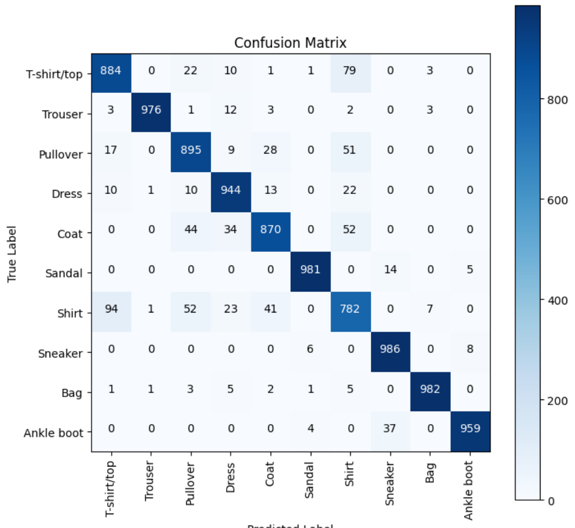
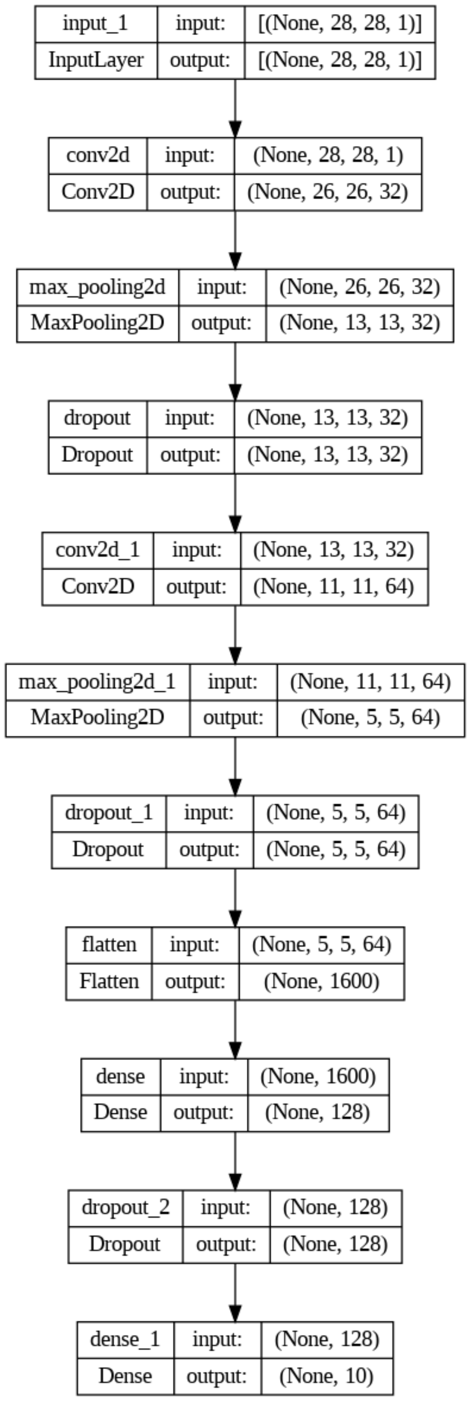
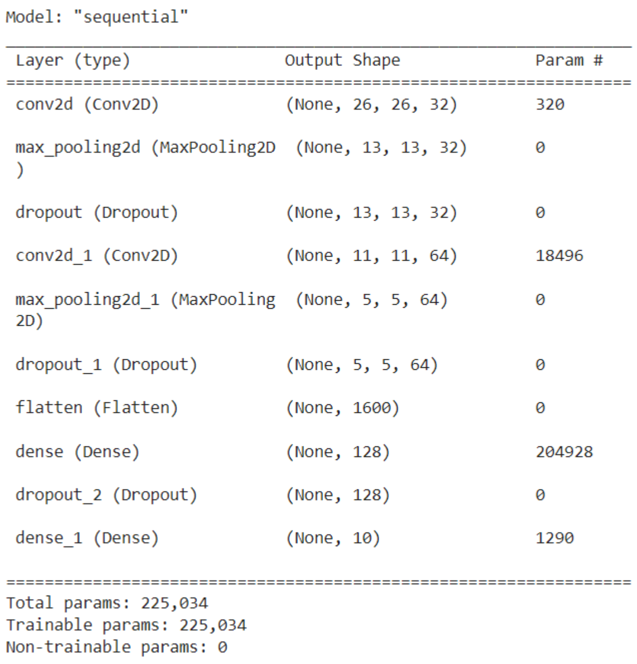


Figure 1. Confusion matrix for best performance model: k-fold & LR & baseline model.

Annex 1: Model Description and Summar


input_1 | input: | [(None, 28, 28, 1)]
InputLayer | output: | [(None, 28, 28, 1)]
conv2d | input: | (None, 28, 28, 1)
Conv2D | output: | (None, 26, 26, 32)
max_pooling2d | input: | (None, 26, 26, 32)
MaxPooling2D | output: | (None, 13, 13, 32)
dropout | input: | (None, 13, 13, 32)
Dropout | output: | (None, 13, 13, 32)
conv2d_1 | input: | (None, 13, 13, 32)
Conv2D | output: | (None, 11, 11, 64)
max_pooling2d_1 | input: | (None, 11, 11, 64)
MaxPooling2D | output: | (None, 5, 5, 64)
dropout_1 | input: | (None, 5, 5, 64)
Dropout | output: | (None, 5, 5, 64)
flatten | input: | (None, 5, 5, 64)
Flatten | output: | (None, 1600)
dense | input: | (None, 1600)
Dense | output: | (None, 128)
dropout_2 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_1 | input: | (None, 128)
Dense | output: | (None, 10)



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 dropout (Dropout)           (None, 13, 13, 32)        0

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 dropout_1 (Dropout)         (None, 5, 5, 64)          0

 flatten (Flatten)           (None, 1600)              0

 dense (Dense)               (None, 128)               204928

 dropout_2 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 225,034
Trainable params: 225,034
Non-trainable params: 0
```