سوال اول

a. w1=h1=w2=h2=32, d1=10, k=16, f=9 assuming that s=1 and we have no padding.
w2=(w1 – f + 2p) / s + 1 => 32 = (32 – 9 + 2p) + 1 => p = 4
parameter count: (f * f * d1 * k) + k = 9 * 9 * 10 * 16 + 16 = 12976

b. w2=(w1 – f + 2p) / s + 1 => w2 = (32 – 5 + 0) / 1 + 1 = 28
h2=(h1 – f + 2p) / s + 1 => h2 = 28
output: 28 * 28 * 3

w2=(w1 – f + 2p) / s + 1 => w2 = (32 – 3 + 0) / 1 + 1 = 30
w3=(w1 – f + 2p) / s + 1 => w3 = (30 – 3 + 0) / 1 + 1 = 28
output: 28 * 28 * 3

c. Learning Rate: The amount that the weights are updated during training is referred to as the step size or the “learning rate.” Specifically, the learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0. The learning rate controls how quickly the model is adapted to the problem. Smaller learning rates require more training epochs given the smaller changes made to the weights each update, whereas larger learning rates result in rapid changes and require fewer training epochs.

Batch Size: The batch size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters. The batch size is a number of samples processed before the model is updated.

Epoch: The number of epochs is a hyperparameter that defines the number times that the learning algorithm will work through the entire training dataset. One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters. An epoch is comprised of one or more batches. The number of epochs is the number of complete passes through the training dataset.

d. In FCNNs, each neuron gets its info and inputs from every single neuron before it. So spatial locality is ignored while in CNNs, each neuron gets limited info (not from all previous neurons) and each pixel is processed with respect to what’s around it, not the whole picture.
https://towardsdatascience.com/cnn-vs-fully-connected-network-for-image-processing-8c5b95d4e42f

e. After obtaining features using convolution, we would next like to use them for classification. In theory, one could use all the extracted features with a classifier such as a softmax classifier, but this can be computationally challenging. To address this, first recall that we decided to obtain convolved features because images have the “stationarity” property, which implies that features that are useful in one region are also likely to be useful for other regions. Thus, to describe a large image, one natural approach is to aggregate statistics of these features at various locations. For example, one could compute the mean (or max) value of a particular feature over a region of the image. These summary statistics are much lower in dimension (compared to using all of the extracted features) and can also improve results (less over-fitting). We aggregation operation is called this operation ”‘pooling”’, or

sometimes "'mean pooling"' or "'max pooling"' (depending on the pooling operation applied). Pooling layers provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map

f. Input image is a * a
Using a 7x7 filter once, stride = 1:
new dimension: (a – 7)/1 + 1 = a – 6
params, considering k=1: 7 * 7 * 1 + 1 = 50
number of multiplications: w * h * f * f = (a – 6)^2 * 7^2

Using a 3x3 filter three times, stride = 1:
new dimension: (a – 3) / 1 + 1 = a – 2
(a – 2 – 3)/1 + 1 = a – 4
(a – 4 – 3)/1 + 1 = a – 6 <= We have the same dimension as before (using a 7x7 filter)
params: 3 * 3 * 3 + 3 = 30.
number of multiplications: w * h * f * f = [(a-2)^2 + (a-4)^2 + (a – 6)^2] * 3^2
Considerable increase in parameter and multiplication count, hence faster computation.

سوال دوم

For this question, the model is as follows:

1- Conv2D 7x7, as suggested in the doc
2- Two consecutive inception modules with different dimensions from the link provided in the doc, to learn as much characteristics as it can; with the slight change of using two consecutive 3x3 Conv2D filters instead of one 5x5 one.
3- A flatten layer to have the shape we want
4- A dense layer to make sure the classification is done to 10 classes.

For training the Neural Network, first I used the code provided in the slides without changing the hyperparameters. The result was ok but I thought I could get better ones. I changed the batch size to twice its original value. What happened was that the test results didn't change but the accuracy for the training data increased, which pointed to the model being overfitted, so I changed it back. Changed the epoch count to 5 and then to 7 to get the original results but in less time (the accuracy when using 5 epochs decreased rather significantly.) The result images are attached with this file and the output it as follows:
Accuracy for result = 0.8983
F1 score for result = 0.8877443173356987
The worst 10 predictions are also available in "10 worst.png"