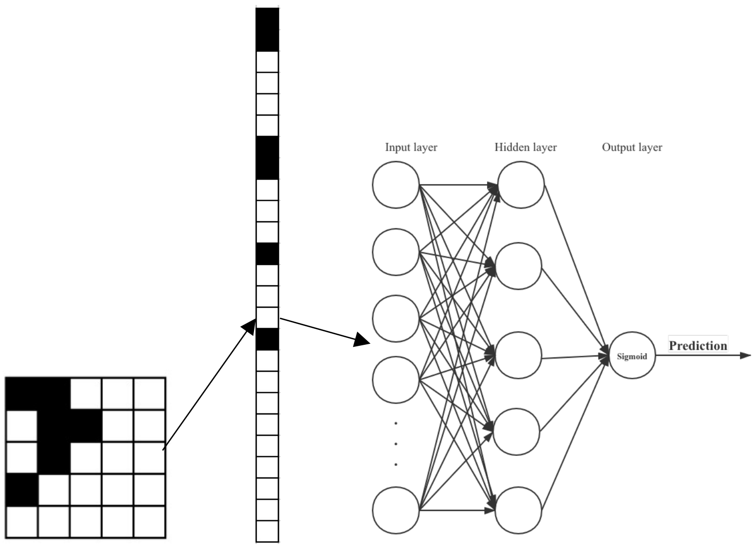
CS520 Final: Question 2 – Classification

Yuange Li (yl1407)

a) My implementation uses a part of the infrastructure code in Assignment 4 to build the model. I choose to use a multilayer perceptron (MLP) model to classify these images. I use the 25 pixels of the 5*5 image as 25 independent input features, so each training image will be represented as a one-dimensional-vector with the length of 25 which only contains 0 or 1. The model has one hidden layer with five hidden neurons. This hidden layer uses Relu() as the activation function. Then the output layer has only one neuron and its output is transformed to a value between 0 and 1, using sigmoid function. Based on this output value of the network we can tag the image as class A if it is lower than 0.5 and tag it as class B otherwise.

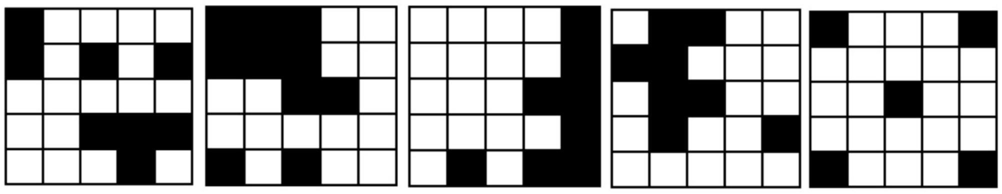
The process and the structure of the network is shown below:



I trained this model using back propagation and gradient descent with Adam optimizer and mean square loss function. The hyperparameters of this neural network is shown below:

Hyperparameter	Learning rate	Epochs	Batch size
Value	0.1	30	10(all training samples per batch)

After training this MLP, I predicted the labels of the given "mystery" images. Here come the results:



B A B A B

As we see the training images intuitively, images in class A are more possible to have more black pixels in the upper left corner and more white pixels in the lower right corner, while images in class B are more possible to have more white pixels in the upper left corner and more black pixels in the lower right corner. Therefore, we can tag the labels of the mystery images artificially based on this rule to evaluate the ability of our model. The second, third and fourth images can be classified obviously since it's very clearly that they belong to class A, B, A respectively according to the distribution of the black pixels. In the first image, there are black pixels in both upper left corner and lower right corner of the image, but the black pixels cover more area in lower right than it does in upper left. So, we can label it as class B. Consequently, the model's predictions make sense on these images. The last image is symmetric, I cannot even classify it so I will not be surprised no matter what result the model returns. The model predicted it as B finally. However, there is a smaller distinction between 0 and the output before the sigmoid layer of this image compared with those on other images. Therefore, the model is not so confident on its prediction of this image.

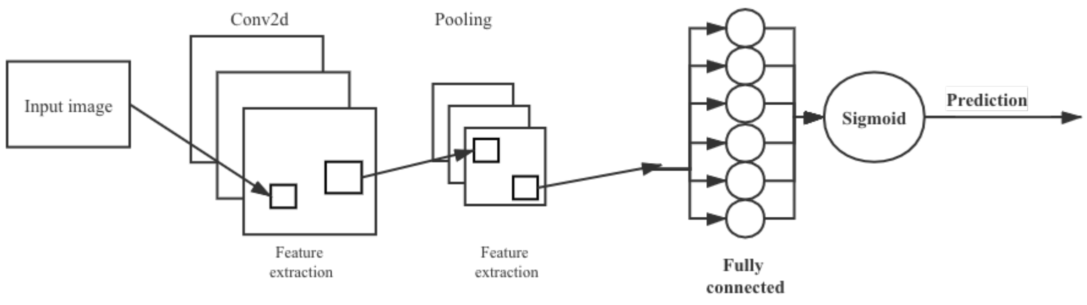
b) First, the data provided are so small so the most efficient way to avoid overfitting is to get more data or to generate more data (for example, draw some images based on the rule by human).

   Second, actually getting more data is impossible. Thus, we can add some noise to the images in the dataset and in each epoch, the noise is supposed to not be the same.

   Third, we can adjust the hyperparameters of the neural network like decrease the learning rate or decrease the steps. The model will stop to fit to the data earlier as a result.

   What's more, we can add dropout to the model and it's an effective way for many neural networks.

c) In the second model, I implement a convolutional neural network since our mission is to classify images. CNN have achieved plenty of great results on many computer vision missions. The structure of the model is as follow. The input is a 5*5 image with only one channel and the shape of the kernel of the CNN is 3*3. Its stride is 1*1 and the output of this conv2d layer has 3 channels. Then, a max pool layer is implemented with pool size of 2*2 and stride of 2*2. At last, there's a fully connected layer followed by a sigmoid activation function to generate the prediction just as the first model.



The predictions of the mystery images given by this model is just the same as the first model which is BABAB. However, the process is much more complex in this model and the results given by this model seem to be more confident based on the output value of the last two layers compared with MLP. The difference between the degree of confidence is because of the different principles of these two models. In the CNN model, the neural network has the ability to learn based on the adjacent pixels while MLP doesn’t. MLP can just process the inputs as separated independent features. The CNN can extract the features based on the relationship of adjacent pixels. Thus, CNN can converge faster than MLP and give more “confident” results compared with MLP. Of course, the cost is that it would spend more time learning based on its larger computation scale.