This HW covers units 2, 3, 4, and 5.

The grade for this homework will be based on the best 3 out of 5 questions. We recommend that you attempt all 5 questions, however only the best 3 will be used, each with a weight of 33.33%.

Question 1: In this question you will implement the training of a two layer network from first principles (without using a library such as Keras). Your network should classify 784 dimensional vectors into one of ten classes. Hence it should work for datasets such as MNIST which are based on 28×28 images and have 10 label types.

For the network architecture, assume there are 100 neurons in the first hidden layer and 10 neurons in the second layer (which is also the output layer). Assume a sigmoid activation function for the first layer, and 10 neurons in the second layer (which is the output layer) operated on with a softmax. Assume cross entropy loss for categorical data. Use mini-batches of size 20, and train using simple gradient descent with a learning rate $\eta = 0.005$.

Implement the code for this network and use your code on the MNIST dataset where for training you should use the first $10,000$ images of the training set and for validation you should use the $5,000$ images that follow. Do not shuffle the dataset.

(a) Train the network for a variable number of epochs until you believe convergence is obtained. Plot the loss, and accuracy both for the training set and the validation set.

(b) Determine the accuracy of your model on the test set ($10,000$ images) and present the confusion matrix.

Reminder: MNIST comes with $60,000$ images that are 'training' (and you will use the first $15,000$ of these for training and validation). It also comes with an additional $10,000$ images that are 'test' and these are the test set you use for (b).

Question 2: Our goal is to implement the ADAM and RMSProp optimization algorithms and compare their performances for the following *Branin* function,

$$f(\theta) = \left(\theta_2 - \frac{5.1}{4\pi^2}\theta_1^2 + \frac{5}{\pi}\theta_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(\theta_1) + 10.$$

First, consider the ADAM algorithm with the following updates:

$$\begin{aligned}
\mathbf{v}^{(k+1)} &= \gamma_v \mathbf{v}^{(k)} + (1-\gamma_v) g^{(k)}, \quad \text{(biased momentum)}\\
\mathbf{s}^{(k+1)} &= \gamma_s \mathbf{s}^{(k)} + (1-\gamma_s)\left(g^{(k)} \odot g^{(k)}\right), \quad \text{(biased squared gradient)}\\
\widehat{\mathbf{v}}^{(k+1)} &= \frac{\mathbf{v}^{(k+1)}}{1-\gamma_v^k}, \quad \text{(unbiased momentum)}\\
\widehat{\mathbf{s}}^{(k+1)} &= \frac{\mathbf{s}^{(k+1)}}{1-\gamma_s^k}, \quad \text{(unbiased squared gradient)}\\
\theta^{(k+1)} &= \theta^{(k)} - \alpha \frac{\hat{\mathbf{v}}^{(k+1)}}{\epsilon + \sqrt{\hat{\mathbf{s}}^{(k+1)}}}, \quad \text{(next iterate)},
\end{aligned}$$

where $g^{(k)} = \nabla f(\theta^{(k)})$ and $\odot$ denotes the element-wise multiplication of two vectors. Note also that division and square root operations between and on vectors are element-wise.

Also, consider the RMSProp algorithm with the following updates:

$$\begin{aligned}
\mathbf{s}^{(k+1)} &= \gamma_s \mathbf{s}^{(k)} + (1-\gamma_s)\left(g^{(k)} \odot g^{(k)}\right),\\
\theta^{(k+1)} &= \theta^{(k)} - \alpha \frac{g^{(k)}}{\epsilon + \sqrt{\mathbf{s}^{(k+1)}}}.
\end{aligned}$$

(a) Is RMSProp a special case of ADAM? Explain.

(b) Using Python, R, or Julia, implement the ADAM algorithm from first principles. Use parameters $\alpha = 0.001$, $\gamma_v = 0.9$, $\gamma_s = 0.999$, and $\epsilon = 10^{-8}$. Take the total number of iterations $N = 2 \times 10^4$ and initialize $\mathbf{v}^{(1)} = \mathbf{0}$ and $\mathbf{s}^{(1)} = \mathbf{0}$.

Report $\theta^{(N)}$, $\|\nabla f(\theta^{(N)})\|$, and $f(\theta^{(N)})$ for $\theta^{(1)} = [5, 20]$.

(c) Now implement the RMSProp algorithm with parameters $\alpha = 0.001$, $\gamma_s = 0.9$, and $\epsilon = 10^{-8}$. Take the total number of iterations $N = 2 \times 10^4$ and initialize $\mathbf{s}^{(1)} = \mathbf{0}$.

Report $\theta^{(N)}$, $\|\nabla f(\theta^{(N)})\|$, and $f(\theta^{(N)})$ for $\theta^{(1)} = [5, 20]$.

(d) To understand the sensitivity of the two algorithms to the learning rate parameter α, repeat tasks (b) and (c) by taking the values of α to be $4, 2, 1$, and 0.1. What are your observations?

Question 3: Consider the FASHION-MNIST dataset and use a framework (e.g. Keras etc...) to carry out a short empirical evaluation of the effectiveness of **batch normalization** for the 10-class classification task. For this construct a fully-connected neural network with 3 hidden layers. Set the number of neurons in hidden layers 1, 2, and 3, to be 400, 200, and 100 respectively. In each of these layers use the ReLu activation function. Then layer 4 is the output layer with 10 neurons and softmax activation. Use the cross entropy loss, the ADAM optimizer with a learning rate of 0.01 (other ADAM parameters as defaults), and train with mini-batches of size 200. Do not use dropout or a regularization term. Use random standard normal initilization for the weights and initilize the biases at 0.

Now refer to the architecture above as A and consider an additional architecture which we can refer to as B. The B architecture is just like A, but also has batch normalisation at the output of each of the hidden layers. That is, in B, there is batch normalisation over the 400, 200, and 100 neurones in the hidden layers. Your goal would be to compare architectures A and B.

Split the training data $(60,000$ images) into a small training set of only $12,000$ images and a large validation set of size $48,000$ (this is 80% of the data used for validation). Note that this is not a realistic split for practical situations, but in this exercise it may aid in speeding up convergence.

(a) Now train architectures A and B, 10 times each, where in teach training run you use 100 epochs. The training runs have different initialisations so they vary. This experiment may require up to several hours of computation time. Plot the loss and accuracy for both the training and the validation set for each of the epochs on each of the training runs. Further, consider an average of the runs, or any other comparison model, and present further plots to compare the performance of A (no batch normalisation) v.s. B (batch normalisation). For example, an average of the 10 runs over each epoch, or any other plots you see fit. Draw conclusions about the effectiveness of batch normalisation based on this experiment.

(b) Based on the validation accuracy, choose the best predictor from the 10 runs of architecture A, and the best predictor from the 10 runs of architecture B. Use early-stopping if you wish (but this isn't mandatory). Compare the accuracy of the 'best A' and 'best B' on the test set.

Question 4: In this question we deal with the CIFAR-10 dataset and use a neural network framework of your choice (e.g. Keras) to compare several convolutional neural network architectures for classification. Use the whole training dataset ($50,000$ images) where the first $40,000$ should be taken for training and the last $10,000$ for validation.

Consider several architectures however in all cases consider the input data as having the three input channels (R, G, and B) of 32×32 images. In all cases use softmax for the output layer, and in all cases use the cross entropy for categorical data loss. Further in all cases use the ADAM optimiser with default parameters and a (default) learning rate of 0.01. If poor convergence performance is observed, adjust the learning rate. In all cases use minibatches of size 100. Further, for all activation functions use ReLu.

All architectures start with 3 convolutional layers, denoted C_1, C_2, and C_3. Further there are two fully connected layers denoted F_4, and F_5, followed by the output softmax layer. We do not use batch normalisation or dropout. In all convolutional layers we use a (default) stride of 1. The convolutional layers are then parameterised by the size of the convolution (3 or 5), the number of output channels (4, 6, or 8), the padding (0, 1, or 2). and the presence of 2×2 maxpooling at the output of the layer, or not.

Consider now the following architectures where the 4-tuple for each of C_1, C_2, and C_3 specify the size of the convolution, the number of output channels, the size of the padding, presence of max-pooling, and the number of neurons of F_4 and F_5.

Architecture	C_1	C_2	C_3	F_4	F_5
	(Size, Channels, Padding, Pooling)			# neurons	# neurons
1	(5,8,0,false)	(3,6,0,true)	(3,4,1,true)	120	100
2	(3,6,1,false)	(3,4,1,false)	(3,4,1,true)	80	150
3	(5,8,2,false)	(5,6,2,false)	(3,4,0,true)	120	100
4	(5,8,2,false)	(5,8,2,false)	(3,6,1,true)	80	150

(a) Train models using all of these four architectures and in each case plot the loss and accuracy both for the training set and the validation set. Train until "convergence is observed" based on a criterion of your choice. Use early-stopping to choose the best training parameters for each model, based on the validation accuracy. As with the previous question keep in mind that the computation time of training may be significant.

(b) Now create an ensemble predictor which operates by averaging the output of the four trained models. That is if the output of model i for image x is the probability vector $p_i(x) \in \mathbb{R}^{10}$, then the ensemble predictor for an image x, is $\hat{f}(x)$ with,

$$\hat{f}(x) = \text{argmax}_{\ell=1,\dots,10} \ \left[\frac{1}{4} \sum_{i=1}^{4} p_i(x) \right]_\ell ,$$

where $[u]_\ell$ is the ℓ'th element of the vector u.

(c) Determine the accuracy of the ensemble predictor on the test set and present the confusion matrix.

Question 5: This question does not require any computation/programming. It rather deals with the design of a neural network architecture that can carry out both object recognition and localization. Assume an input image of size 60×60 which for simplicity we assume is monochrome (e.g. black and white). Say we now wish to consider multiple sub-images, each of size 20×20 and for each such sub-image we wish to carry out classification for one of 5 possible labels. We then wish to determine in which sub-image the classification is most probable and the classification value. For example, if the whole image is indexed via $(1{:}60, 1{:}60)$ then the sub-image can be $(1{:}20, 1{:}20)$ (meaning the top-left corner sub-image), or $(41{:}60, 41{:}60)$ (meaning the bottom-right corner), or any other sub-image of size 20×20 such as $(19{:}38, 11{:}30)$ etc..

As an aid, assume we have a trained convolutional neural network, denoted $\mathcal{M}$ which is designed to work on monochrome input images of size 20×20. The network $\mathcal{M}$ outputs a probability vector of length 5 which indicates likelihood of the 5 labels.

A simple recognition and localization model, $\mathcal{R}_1$ then uses $\mathcal{M}$ by applying $\mathcal{M}$ repeatedly on all possible sub-images with a stride of 2. This means applying it on the $(1{:}20, 1{:}20)$ sub-image, the $(3{:}22, 1{:}20)$ sub-image, the $(5{:}24, 1{:}20)$ sub-image, and moving all the way to the $(41{:}60, 41{:}60)$ sub-image. This is a total of $21 \times 21 = 441$ applications of $\mathcal{M}$ on sub-images which results in 441 output probability vectors. The recognition and localization model $\mathcal{R}_1$ then picks the highest probability from all these vectors (breaking ties arbitrarily). If this highest probability is p_{ij} where i is the index of the sub-image ($i = 1, \dots, 441$) and j is the index of the label ($j = 1, \dots, 5$), the model then concludes that in the input image, one can find label j in the sub-image indexed by i.

It turns out, that $\mathcal{R}_1$ can be improved to a model $\mathcal{R}_2$ which uses a single (bigger) convolutional network only once. Denote this network via $\tilde{\mathcal{R}}_2$. The network $\tilde{\mathcal{R}}_2$ operates on the whole 60×60 input image and produces a $21 \times 21 \times 5$ tensor, say P, which contains the same probability vectors which we would observe if applying $\mathcal{M}$ to each of the 21×21 sub-images separately (as is done with $\mathcal{R}_1$). In the output tensor P, the entry $P_{i_1,i_2,j}$ indicates the probability of having label j in the image associated with the sub image defined by the index (i_1, i_2). Specifically, the 5-vector $P_{i_1,i_2,\cdot}$ is a probability vector associated with the output of $\mathcal{M}$ of the sub-image.

The recognition and localization model $\mathcal{R}_2$ then applies $\tilde{\mathcal{R}}_2$ to the 60×60 input images and concludes both the location and label based on the maximal entry in P.

The construction of a model like $\mathcal{R}_2$ and the neural network $\tilde{\mathcal{R}}_2$ based on a neural network like $\mathcal{M}$ is described well in this video by Andrew Ng. However the notation that we use in this question is not used in the video and the dimensions are different. Watch the video.

Now assume that $\mathcal{M}$ has a given architecture, and describe in detail how to create the neural network $\tilde{\mathcal{R}}_2$ based on $\mathcal{M}$. Describe the dimensions of $\tilde{\mathcal{R}}_2$ and specify how to use the (trained) parameters of $\mathcal{M}$ to construct $\tilde{\mathcal{R}}_2$.

Assume the following as the architecture of $\mathcal{M}$.

Layer 1: 5×5 convolutional, padding $= 2$, stride $= 1$, 8 output channels, no pooling.

Layer 2: 3×3 convolutional, padding $= 1$, stride $= 1$, 16 output channels, 2×2 pooling.

Layer 3: 3×3 convolutional, padding $= 0$, stride $= 1$, 12 output channels, 2×2 pooling.

Layer 4: Fully connected layer with 50 neurons.

Layer 5: Fully connected layer with 100 neurons.

Output layer: 5 neurons with softmax.