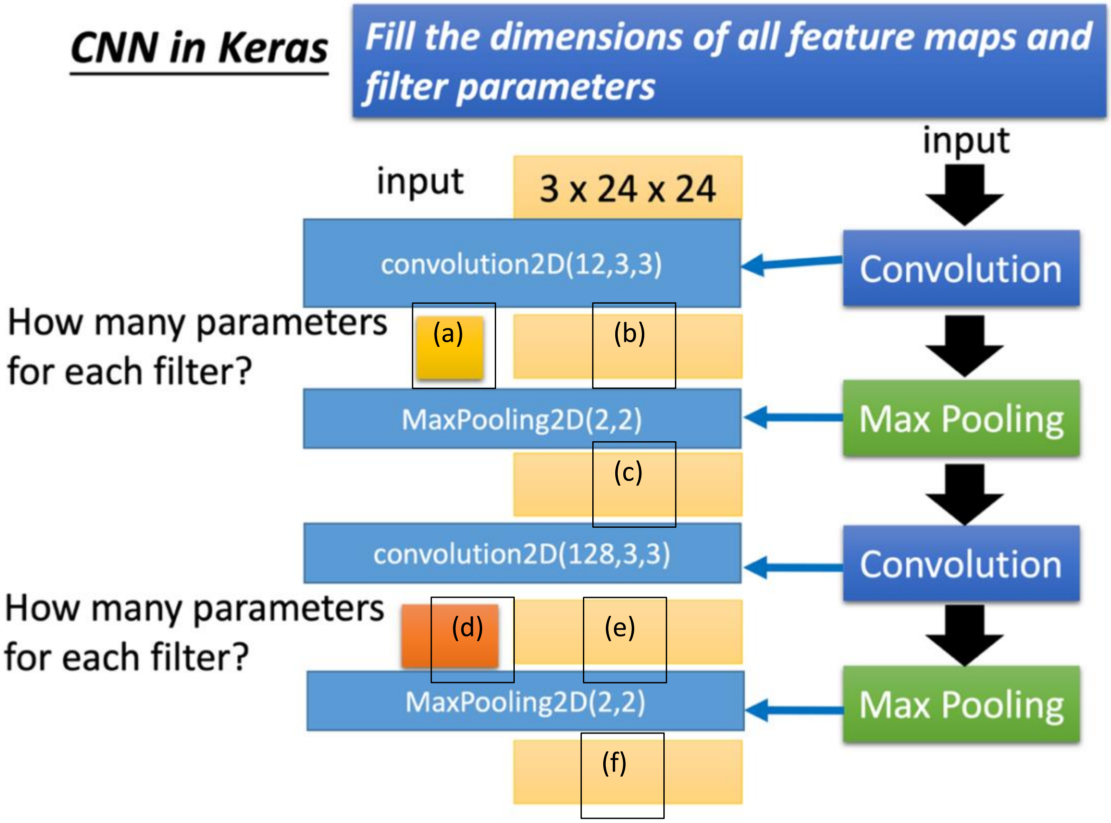
Question1: Dimension reduction and visualization (10 points)

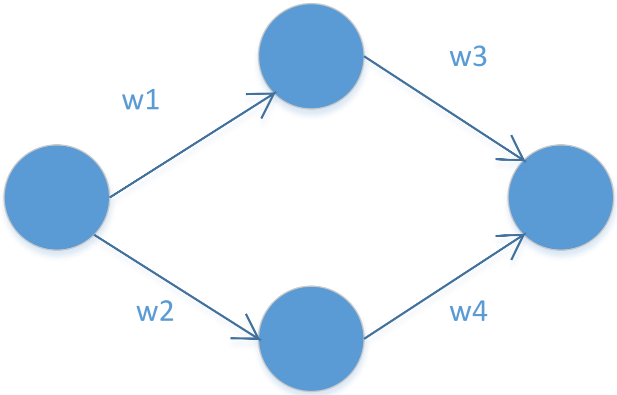
(a) Traditional multi-layer feedforward perceptron ANN has a notorious difficulty to train for models with multiple hidden layers, which is partially caused by the gradient vanishing issue. Describe what is the gradient vanishing problem and how the recent deep learning algorithms address this issue.

(b) For the following CNN network, calculate the feature map dimensions and the parameter numbers for the filters. We use no padding and the stride is 1.



Question 2: Back-propagation algorithms (10 points)

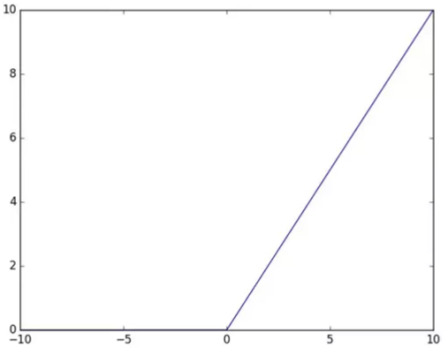
Derive the back-propagation algorithm for the below Neural network using two different activation functions.



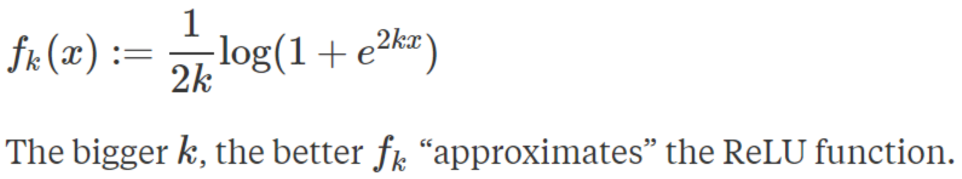
(1) For the above multi-layer neural network, derive the formulas of update rule to train the weights w1, w2 and w3, w4 using the idea of back-propagation using the traditional activation function of the neurons, the sigmoid function S(x).

In deep neural networks, the most frequently used activation function is ReLU, defined as f(x):=max(0,x). This means that f is 0 when x≤0 and equal to x when x≥0

Here is how it looks like:



The gradient computation for this function is thus straightforward everywhere except at 0: 1when x>0and 0 when x<0. The only issue of this is that there is no definition of gradient at x=0. Sometimes, people uses the following approximation function to replace it. when you set k=1 or 2. It is very close to the above function.



$$f_k(x) := \frac{1}{2k}\log(1 + e^{2kx})$$

The bigger k, the better f_k "approximates" the ReLU function.

(2) Derive the formulas of the update rule to train the weights w1, w2 and w3, w4 using this new activation function above.

(3) Describe how the Sigmoid function as activation functions can lead to gradient diminishing issue for conventional multi-layer Perceptron neural network?

(4) Describe the relationship of two hyper-parameters of DNN network training: the learning rate and the batch size

(5) Read the below two blog on hyperparameter tuning and summarize three key points

https://towardsdatascience.com/hyper-parameter-tuning-techniques-in-deep-learning-4dad592c63c8
https://bolster.ai/blog/three-lessons-learned-while-tuning-hyper-parameters-of-deep-learning-models/

Question 3: Do a research and describe what are the key progresses of deep neural networks compared to previous conventional multi-layer Perceptron (feedforward neural networks). (5 points)

Question 4: Deep Learning classification using a deep neural network package. (15 points)

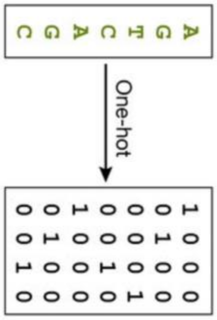
Study the sample MNIST digit recognition code https://github.com/pytorch/examples/tree/main/mnist and play with it.

Then try to develop a neural network to classify the provided human DNA sequences and random ATCG sequences (humanvsran.seq.txt and humanvsran.label.txt), using these deep neural network models.

Develop two deep neural network models: one is a regular fully connected feedforward BP network (with 2-3 hidden layers), another is a convolutional neural networks with multiple convolution layers + fully connected layers

- You need to randomly shuffle the 20,000 records and then pick the first 15000 as training samples and the remaining 5000 as test samples.
- Report the average accuracies of 10 runs with random splitting of both models on the problem.
- You need to draw the DNN models showing its layers and number of nodes in each layer following this example (https://github.com/yu4u/convnet-drawer)

You need to use the one-hot encoding to convert a DNA sequence into a matrix as the sample to the neural network.



Question 5: Regression using deep neural networks (10 points)

The attached four csv files are the data files for superconductor critical temperature prediction problem. (The last column is the temperature to be predicted, the rest columns are features) . Write a deep neural network model for regression, and apply it to the below 3 datasets. Report the average R2 regression score for 5 runs, in each of which the whole dataset is randomly split by 80% for training and 20% for testing. Try to tune the architecture of the DNN network to improve the results. (You can change the no. of layers, the activation function, adding batch normalization, adding dropout layer, using convolution layers,

Dataset	**Deep neural network1**	**Deep neural network2**	**Deep neural network3**	
	Features of this model	**Features of this model**	**Features of this model**	
DataK	**(fill R2 score..)**			
DataV				
Superconductor.csv				