EE708: Fundamentals of Data Science and Machine Intelligence

Assignment 6

Based on Module 7: Deep Neural Networks, Convolutional Neural Networks, Recurrent Neural Networks, and Autoencoders

1. Why is it generally preferable to use a logistic regression classifier rather than a classic perceptron (i.e., a single layer of threshold logic units trained using the perceptron training algorithm)? How can you tweak a perceptron to make it equivalent to a logistic regression classifier?
2. How many neurons do you need in the output layer if you want to classify email into spam or ham? What activation function should you use in the output layer? If instead you want to tackle MNIST, how many neurons do you need in the output layer, and which activation function should you use?
3. Suppose you have an MLP composed of one input layer with 10 passthrough neurons, followed by one hidden layer with 50 artificial neurons, and finally one output layer with 3 artificial neurons. All artificial neurons use the ReLU activation function.
   a. What is the shape of the input matrix **X**?
   b. What are the shapes of the hidden layer's weight matrix $\mathbf{W_h}$, and bias vector $\mathbf{b_h}$?
   c. What are the shapes of the output layer's weight matrix $\mathbf{W_o}$, and bias vector $\mathbf{b_o}$?
   d. What is the shape of the network's output matrix **Y**?
   e. Write the equation that computes the network's output matrix **Y** as a function of $\mathbf{X}$, $\mathbf{W_h}$, $\mathbf{b}_\mathrm{h}$, $\mathbf{W_o}$ and $\mathbf{b_o}$.
4. Show the derivative of the error function eq. (1) with respect to the activation a_k for an output unit having a logistic sigmoid activation function satisfies eq. (2).
   If we consider a training set of independent observations, then the error function, which is given by the negative log likelihood, is then a *cross-entropy* error function of the form

$$E(w) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} \qquad (1)$$

   Where y_n denotes $y(\mathbf{x}_n, \mathbf{w})$. Note that there is no analogue of the noise precision β.

$$\frac{\partial E}{\partial a_k} = y_k - t_k \qquad (2)$$

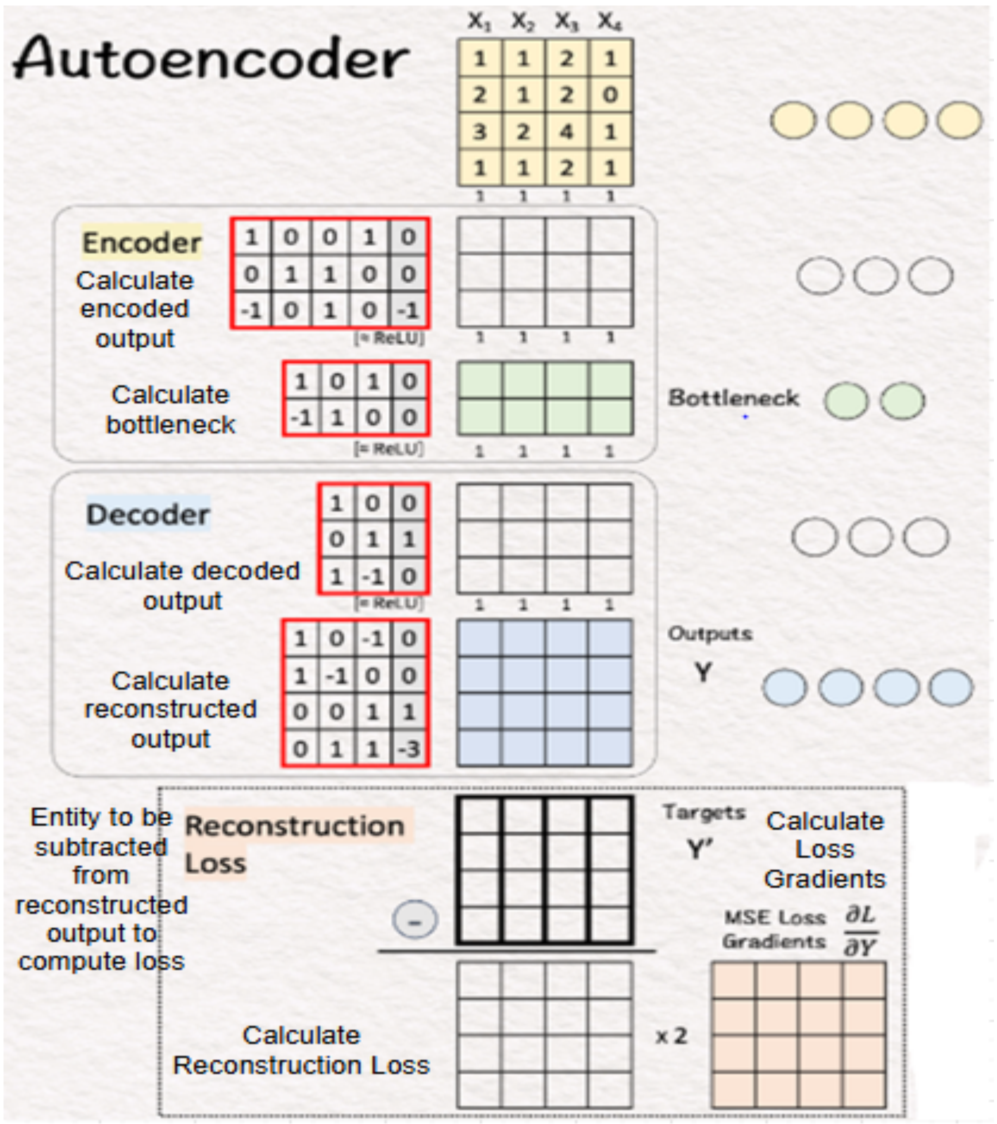
5. Explicitly calculate the output of the following convolution of a 1 × 4 input matrix with a 2 × 2 filter:

$$\begin{array}{|c|c|c|c|}\hline 2 & 5 & -3 & 0 \\ \hline 0 & 6 & 0 & -4 \\ \hline -1 & -3 & 0 & 2 \\ \hline 5 & 0 & 0 & 3 \\ \hline \end{array} \;*\; \begin{array}{|c|c|}\hline -2 & 0 \\ \hline 4 & 6 \\ \hline \end{array} \;=\; \begin{array}{|c|c|c|}\hline ? & ? & ? \\ \hline ? & ? & ? \\ \hline ? & ? & ? \\ \hline \end{array}$$

6. Consider a one-dimensional strided convolutional layer with an input having four units with activations (x_1, x_2, x_3, x_4), which is padded with zeros to give $(0, x_1, x_2, x_3, x_{4,}0)$, and a filter with parameters (w_1, w_2, w_3). Write down the one-dimensional activation vector of the output layer assuming a stride of 2. Express this output in the form of a matrix **A** multiplied by the vector $(0, x_1, x_2, x_3, x_4, 0)$. Now consider an up-sampling convolution in

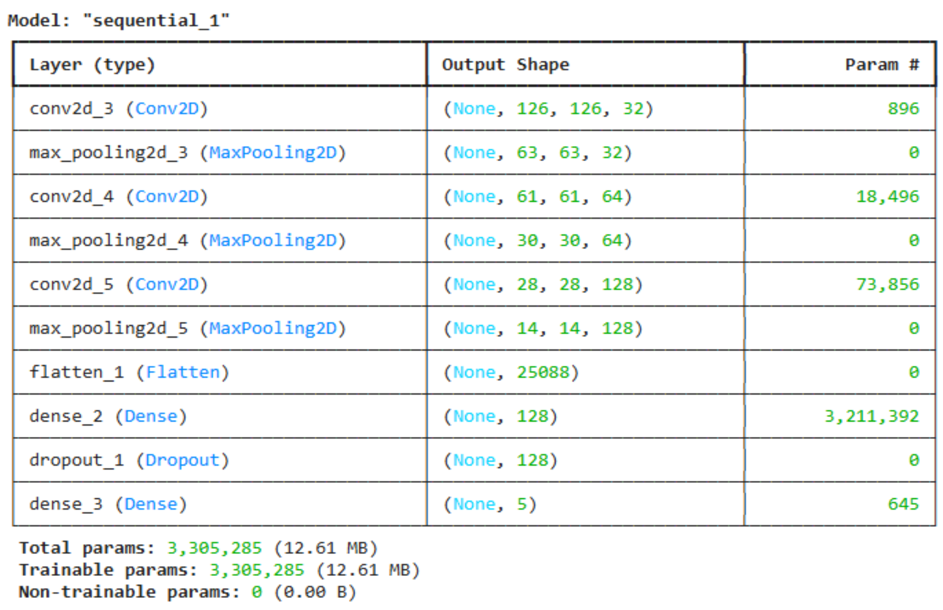
which the input layer has activations (z_1, z_2) with a filter having values (w_1, w_2, w_3) and an output stride of 2. Write down the six- dimensional output vector assuming that overlapping filter values are summed and that the activation function is just the identity. Show that this can be expressed as a matrix multiplication using the transpose matrix $\mathbf{A}^{\mathbf{T}}$.

7. For a simple autoencoder model as shown below, work through the complete forward pass. Fill in the missing values by calculating the encoded (latent) outputs, decoded outputs, and the final reconstructed output. Then, compute the reconstruction loss and determine the gradient of the loss. Ensure all steps are shown clearly with proper calculations. ReLU activation function is used in model.



Programming Questions:

1. Image classification using Convolutional Neural Network (CNN):
   - Load *tf_flowers* dataset from *tensorflow_datasets*.
   - Split data into train and test sets with an 80:20 ratio and normalize pixel values.
   - Build a CNN Model with the following summary:



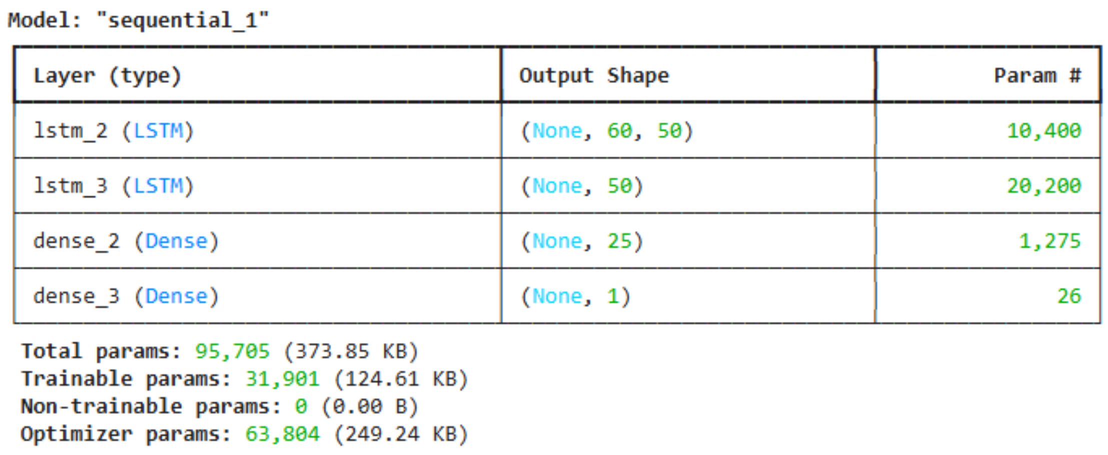
```
Model: "sequential_1"
┌─────────────────────────────────┬────────────────────────┬───────────────┐
│ Layer (type)                    │ Output Shape           │       Param # │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_3 (Conv2D)               │ (None, 126, 126, 32)   │           896 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_3 (MaxPooling2D)  │ (None, 63, 63, 32)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_4 (Conv2D)               │ (None, 61, 61, 64)     │        18,496 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_4 (MaxPooling2D)  │ (None, 30, 30, 64)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_5 (Conv2D)               │ (None, 28, 28, 128)    │        73,856 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_5 (MaxPooling2D)  │ (None, 14, 14, 128)    │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten_1 (Flatten)             │ (None, 25088)          │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                 │ (None, 128)            │     3,211,392 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_1 (Dropout)             │ (None, 128)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_3 (Dense)                 │ (None, 5)              │           645 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 3,305,285 (12.61 MB)
 Trainable params: 3,305,285 (12.61 MB)
 Non-trainable params: 0 (0.00 B)
```

- Use ReLU activation for hidden layers and Softmax activation for the output layer.
- Train the model using Adam optimizer and sparse categorical cross-entropy loss for 10 epochs and evaluate on the test set.

a. Plot training and testing accuracy over epochs.

b. Generate the classification report and confusion matrix.

2. Stock price prediction using Recurrent Neural Networks (RNNs), use dataset *A6.csv*:
   - Normalize the stock prices and create sequences of 60 days of stock prices as input and the next day's price as the target.
   - Split data into train and test sets with an 80:20 ratio.
   - Build an RNN model using LSTM layers with the following summary:



```
Model: "sequential_1"
┌─────────────────────────────────┬────────────────────────┬───────────────┐
│ Layer (type)                    │ Output Shape           │       Param # │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ lstm_2 (LSTM)                   │ (None, 60, 50)         │        10,400 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ lstm_3 (LSTM)                   │ (None, 50)             │        20,200 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                 │ (None, 25)             │         1,275 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_3 (Dense)                 │ (None, 1)              │            26 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 95,705 (373.85 KB)
 Trainable params: 31,901 (124.61 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 63,804 (249.24 KB)
```

   (first LSTM layer with *return_sequences=True*)

   - Use Mean Squared Error (MSE) as the loss function and train the model for 20 epochs with a batch size 32.
   - Predict stock prices on the test set and inverse transform the values to the original scale. Plot the actual and predicted stock prices.