SFU SCHOOL OF MECHATRONIC SYSTEMS ENGINEERING

Simon Fraser University
School of Mechatronic Systems Engineering
MSE491 - Application of Machine Learning in Mechatronic Systems

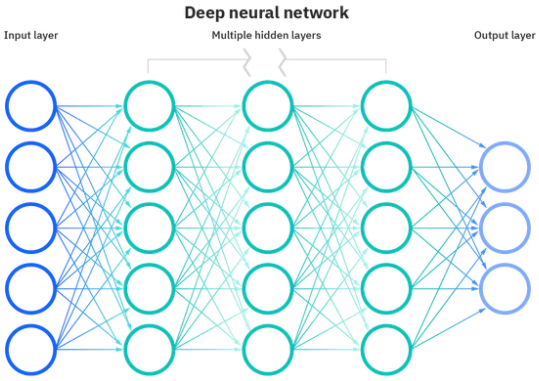
Lab 3 – Neural Networks

Due Date: Mar 28st, 11:59pm

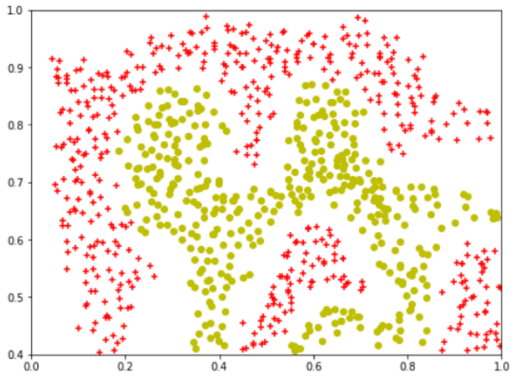
Background

Artificial Neural Networks (ANNs) consist of a number of artificial neurons (or nodes) that resemble the behaviour of the human brain neurons. These networks allow computer programs to learn and recognize complex patterns in the data. ANNs are comprised of three main layers, the input layer, a single or multiple hidden layers, and an output layer (Fig.1). The nodes are connected to each other with an specific weight and threshold. The weights for the connections get adjusted after training the ANN. The nodes get activated if the output of an individual node is higher than the threshold.

One common application to the ANNs is to solve complex nonlinear regression and classification problems (e.g. Fig.2). Multiple training algorithms are available for ANNs. Other advantages of ANNs including their ability to implicitly detecting complex nonlinear relationships, detecting all possible interactions between predictor variables.



©https://www.ibm.com
Fig1. ANNs schematic



© https://towardsdatascience.com
Fig2. An example for complex classification problem

However, there are a number of challenges that exist working with ANNs. These challenges include the "black-box" nature of the ANNs, greater computational cost, and proneness to overfitting. Both advantages and challenges should be taken into account while implementing ANNs to solve real-world mechatronics problems.

Recent advances in Computer Vision with Deep Learning revealed the efficiency of one particular algorithm, called Convolutional Neural Network (CNN) for image processing. A CNN is a Deep Learning algorithm that is able to capture the spatial and temporal dependencies in an image by implementing relevant filters. Since CNN reduces the number of parameters involved and reusability of weights, the architecture performs a better fitting to the image dataset. Please review the contents presented in this link to learn more about the details of CNNs.

MSE 491- Assignment 3 (Spring 2022)

Image Classification

Rock-Paper_Scissors Data Set from Tensorflow

This dataset contains images of hand playing rock, paper, scissors game. This includes hand from different races, ages and genders, posed into Rock, Paper, or Scissors and labelled as such. Each image is 300 by 300 pixels in 24-bit color. You can read more information on this dataset here. The main goal of this assignment is to build a classifier to classify the three different hand poses using the images and labels provided in the training set. The images and labels in the test set will be used to evaluate the trained model.

For this assignment, please follow the instructions given in this document to complete and run different sections in the .ipynb file provided along with the assignment . You can open and run the .ipynb file either in Jupyter notebook or Google Colab.

1. Getting Started

Open Assignment3.ipynb in Jupyter notebook or Google Colab. Run the first command to install the requirements for the dataset. Then, import the necessary libraries using the commands provided in the next section. You can view the list of available tensorflow datasets running "tfds.list_builders" in the next section.

1. Using "tfds.builder", choose the "rock_paper_scissors" as your builder. Print out the information on this dataset in the .ipynb file and answer the following questions:
   a. What is the image shape? What does each number mean? **(3 points)**
   b. What percentage of the data is considered as the training set? **(2 points)**
2. Load the data using "tfds.load" and split the train and test set.
   c. Print out three example images from the test dataset. **(2 points)**
3. Convert the tensorflow dataset format into numpy format. Adjust the number of colour channels to 1 so that the images are considered as grey-scaled. When using Neural Networks in Keras, there is always need to include a colour channel. Considering that the images are grey scaled, re-shape the train and test data by adding the number of colour channels (i.e. 1) at the end **(3 points).**
4. Scale the values in the two-dimensional matrix of image pixels between 0 and 1 **(3 points)**.
5. Select a random image. Print out the the two-dimensional matrix values of image pixels. Explain why the range is limited to certain values **(2 points)**.

2. Basic Feed-Forward Neural Network Model

Create a sequential 3-layer model (Fig.3) using "keras.layers.Dense" considering the following instructions:

1. Flatten the data using keras.layers.Flatten().
2. Choose "relu" as the activation function for the first two layers (i.e. input and hidden layer) and "softmax " as the activation function for the last layer (i.e. output layer) **(5 points)**.
3. Use "model.compile" to compile the 3-layer ANN. Choosing "adam" as the optimizer, loss function as "Sparse Categorical Crossentropy", and pass in the "metrics" as "accuracy" **(5 points)**.
4. Fit the data into the model using "model.fit". Choose the number of epochs as 10 **(5 points)**.
5. What is the last epoch's accuracy? Evaluate your model on test data using "model.evaluate(test_images, test_labels)" and discuss the results. Provide a line plot (epoch number vs. accuracy) to support your discussions. **(10 points)**.
6. Choose "tanh" as the activation function for the first two layers and repeat steps 3-6 (this would be a new model using different activation function). Using bar charts, show how do the evaluation results (i.e. loss and accuracy) change? Discuss why **(5 points).**
7. Is there any advantage for using "softmax" as the activation function for the output layer? **(5 points)**

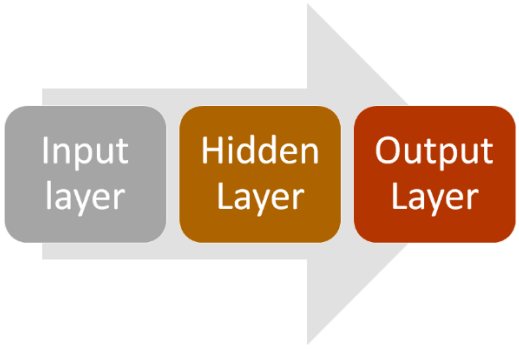


Fig3. Schematic of the first feed forward network

3. CNNs

A. **First CNN**

Create a sequential 4-layer model (Fig.4) following the instructions below to build your first CNN:

1. For the first two layers (i.e. the input layer and the first hidden layer) use "keras.layers.Conv2D" with kernel size of 3 to create the layers with "relu" activation function **(5 points)**.
2. To pass the output of these two layers to the output layer, flatten the data in a flattening layer.
3. Using "keras.layers.Dense", create the output layer with "softmax " as the activation function.
4. Compile the model as in section 2.
5. Fit the data into the model using "model.fit". Choose the number of epochs as 10[1].
6. Evaluate your model on test data and report the accuracy. You can use "model.evaluate" for this purpose. Discuss how it changed from when you used a basic neural network. Is this network performing worse? Why? **(10 points)**

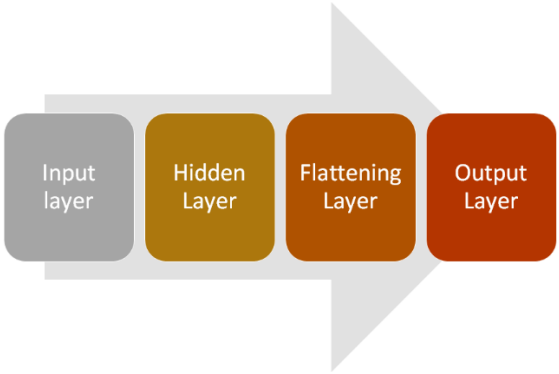


Fig4. Schematic of the first CNN

[1] You can run this much faster if you use GPU. If you do not have access to GPU on your PC, simply use Google Colab and choose the GPU option from the Runtime drop-down menu. Please remember you need to run everything from scratch again if you choose this option in Google Colab.

B. **Second CNN**

Follow the instructions below to modify your first CNN model.

1. Convert the input layer to an average pooling layer. Set the pool size to 6 and stride as 3. Explain how this conversion affect the output of this layer **(5 points)**.
2. Use "keras.layers.Conv2D" with kernel size of 3 and "relu" activation function to create two hidden layers of CNNs (Fig.5).
3. Compile, fit, and evaluate the new model as before. Report the accuracy after evaluating your model **(5 points)**.
4. Change the stride once to 2 and once to 6. Compile, fit and evaluate the results again for both scenarios.
5. Using bar charts, discuss how changing the stride to 2 or 6 change the evaluation results **(5 points)**.

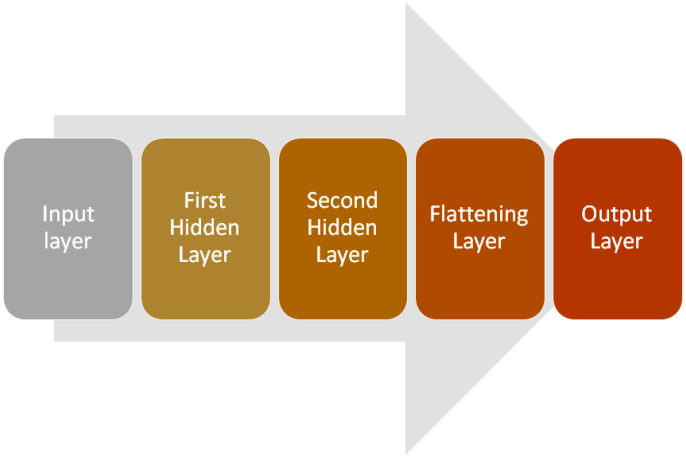


Fig5. Schematic of the second CNN

C. **Third CNN**

Follow the instructions below to modify your second CNN.

1. Add a max-pooling layer after the two hidden layers of CNN. Choose the window size of 2 by 2 (Fig.6).
2. Compile, fit, and evaluate the new model as before. Report the accuracy after evaluating your model **(5 points)**.

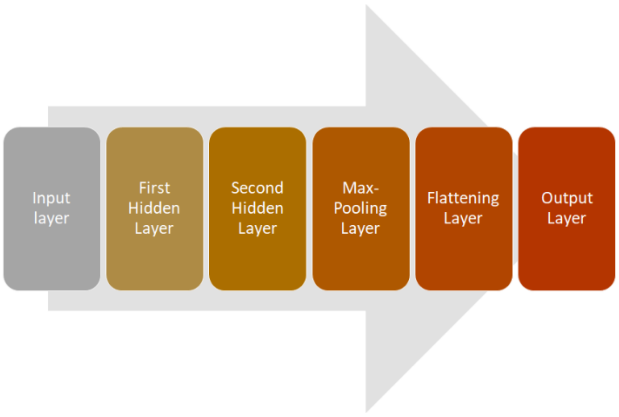


Fig6. Schematic of the third CNN

D. **Fourth CNN**

Follow the instructions below to modify your third CNN:

1. Using “keras.layers.Dropout”, cut out %40 of the connections after maxpooling (Fig.7).
2. Compile, fit, and evaluate the new model as before. Report the accuracy after evaluating your model **(5 points)**.

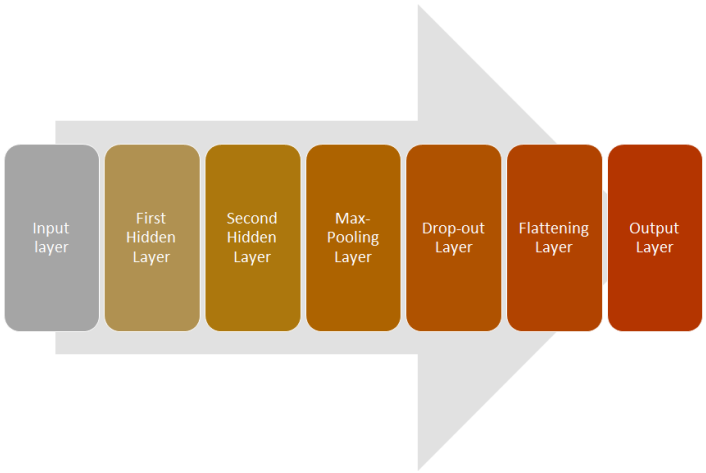


Fig7. Schematic of the fourth CNN

Finally, using bar charts, compare the accuracy of the four CNN models. Discuss the effects of modifications for each sections of A, B, C, and D separately **(10 points)**.

Submitting your Assignment

The assignment must be submitted online at Canvas with the following structure:

For each group, you must submit **one zipped folder** that contains:

1. **A report**- The assignment report must be in **PDF format**, called **report.pdf**. This report must contain all the figures and discussions/explanations for the questions.
2. **An .ipynb file that contains all your codes**- This file must be called **Assignment3.ipynb**. Only **a single file** is acceptable. There is space provided for each section. **Please do not add/remove sections.** Please fill out the sections with your code.