Computational Machine Learning COSC 2673
Assignment 2: Classify Images of Colon Cancer
Group Name: UK

(Karthi Narendrababu Geetha: s3835901, Udita Kapila: s3689968)

Problem Statement:

Classification and detection of cell from the histopathologic images is one of the hardest tasks due to the heterogeneity in its cellular architecture. In this paper, we've discussed our approaches on solving this problem, the evaluations and the results. We have given a dataset of 99 patients out of which 30 patients data does not contain the labels for the type of cancer cell. The dataset consists of 27x27 RGB images of colon cells and we built an end-to-end machine learning systems for the following three tasks:

- Classify images according to whether given cell image represents a cancerous cell or not.
- Classify images according to cell-type, such as: fibroblast, inflammatory, epithelial or others.
- Improving the efficiency of the cell type classification by using the unlabelled dataset.

Approaches:

Task – 1:

- ➢ The task is to predict whether the cell image is cancerous or not.
- ➢ We started with building a basic machine learning model using Stochastic Gradient Descent (SGD) classifier algorithm. We chose this algorithm, because this is the base for the neural network and deep learning algorithm which is considered as the best image classification algorithms in the recent days.
- ➢ Though we got a promising result from this algorithm, we wanted to explore more in Neural networks and hence we trained a classic neural network (CNN) model with 1 hidden layer.
- ➢ The accuracy of the CNN model has not improved and hence we built and trained a Deep CNN model based on VGG architecture (fig. a from Appendix). The architecture of VGG model consists of,
  - 3x3 Convolution kernel layers with filters of 32, 64 and 128.
  - Pooling layers which are always maximum, and they are 2x2.
  - Padding is same and activation is Relu.
  - Kernel regularizer with lambda value of 0.001.

Task – 2:

- ➢ The task is to predict images based on cell types, such as fibroblasts, inflammatory cells, epithelial cells, or other cells.
- ➢ Similar to task – 1, we initially develop a classification model using SGDClassifier, however we haven't got much accuracy.
- ➢ Next, we tried building a CNN model with 1 hidden layer and trained it. Our accuracy has improved for around 10%, but we wanted to explore more using Deep learning algorithms.
- ➢ And so we built 2 different types of deep learning models.

VGG:

VGG model's architecture is similar to the one which we used in the previous task (Task - 1).

LeNet model:

LeNet model's architecture consists of,

- 5x5 & 3x3 Convolution kernel layers with filters of 32, 48 & 64.
- Pooling layers which are always maximum with size 2x2 and strides 2.
- Hidden layers with units 256 & 84.
- Padding is same and activation is Relu.
- Kernel regularizer with lambda value of 0.001.

Task – 3:

As the medical experts have provided the type of cancerous cell only for the first 60 patients, the 30 patients data is unlabelled. Anyway, in order to improve the accuracy of our model, we need to make use of all the data. Hence, we came up with an approach which is a combination of the **Semi-Supervised learning** and the **Transfer Learning**.

Algorithm:

- ✓ **Step – 1:** Load the main labelled data.
- ✓ **Step – 2:** Train the model in order to predict the cellType of the cancerous cell.
- ✓ **Step – 3:** Save the model.
- ✓ **Step – 4:** Load the unlabelled data and split it into 3 equal halves.
- ✓ **Step – 5:** Load the first batch of data.
- ✓ **Step – 6:** Predict the cellType using the previously saved model **from Step – 3.** (Assuming that the predicted values are correct)
- ✓ **Step – 7:** Append the first batch of data with the main data and store it as a separate variable.
- ✓ **Step – 8:** From the previously saved model, deactivate the trainable layers and fit the model again.
- ✓ **Step – 9:** Save the model.
- ✓ **Step – 10:** Repeat **Steps 6 – 9** for the next 2 batches of data, except for the **Step – 7**.
- ✓ **(Repeat) Step – 7:** Append the batches of data with the previously appended variable.

Data Augmentation:

In order to fine tune the model for better accuracy, data augmentation is done which increases the number of copies of data for the training model. For our training model to achieve better accuracy, we've done the following process of data augmentation.

- Set the rescaling factor to **1./255.**
- Set the rotation range of **30 degrees**.
- Set the width & height shift ranges to **0.2**, which will shift the pixels of images.
- Set the zoom range to **0.2**.
- Set the normalizations to True.

Training Neural Network Models:

All the models that we used are trained with the epochs of 25, 50, 100 & 150. The training accuracy and loss graphs of each model is in the appendix section.

Independent Evaluation (Comparing our result with other's research work):

We have taken a research paper which was **published on Sensors** and topic was "A Machine Learning Approach to Diagnosing Lung and Colon Cancer Using a Deep Learning-Based Classification Framework

Comparison of Model Architecture and Classification Algorithm:

In this research paper, LC25000 dataset was used which contains 25,000 colour-images of five types of lung and colon tissues and 70% of the images chosen to train this model and the remaining 30% image to test it. After data pre-processing and image Augmentation, Image dimensions was 64x 64. CNN model was used to classify the histopathological cancer images which had three convolution layers, two max-pooling layers, a batch-normalization layer, and a dropout layer. Epoch =500 with batch size=64 and relu activation function was used with 30% Dropout.

Our dataset includes 27x27 RGB images of colon cells from 99 different patients. The output of the algorithm is one of four types of cells - fibroblasts, inflammatory, epithelial, or others. We have used VGG model with 3x3 Convolution kernel layers with filters of 32, 64 and 128 and 2x2 Pooling layers and Padding is same and activation is relu and kernel regularizer with lambda value of 0.001 with 20% dropout.

We have used ImageDataGenerator to perform data conversion on the image. With ImageDataGenerator, any type of transformation can be applied to the image passed to the model.

Result:

From research paper they achieved 96.33% peak classification accuracy and assures that the model is highly accurate and reliable (96.38% F-measure score) for lung and colon cancer identification.

We have applied four models in our dataset and out of which we got the highest accuracy with VGG model with test accuracy of 77.4% and training Accuracy of around 78%.

```
Accuracy score:  0.7742424242424243
F1 score:  0.7296347164149486
```

Comparing Both Models:

The task of our first model is to predict whether images are cancerous or not. We have started with traditional supervised learning algorithm Stochastic Gradient Descent (SGD) classifier algorithm. We got 71% accuracy with hyper tunning and tried Classic Neural network model for better accuracy and achieved 87% Test accuracy and we further tried VGG for much better accuracy but due to overfitting we had to do hyperparameter tunning using regularisation and Augmentation techniques and we finally achieved best test accuracy of 88% with f1-score=0.87

Our second task was to predict the images based on cell types where we had started with Stochastic Gradient Descent (SGD) classifier algorithm and achieved 61% accuracy and with fine tuning we got 62% accuracy. For better accuracy we tried classic neural network and got 70% accuracy. To further improve this accuracy, we tried Lenet model and with some tuning we got 75% accuracy and finally we applied VGG model and initially we got around 75% with overfitting. So, we applied Regularisation and Augmentation to overcome this problem and achieved 77% test accuracy with 72% f1-score.

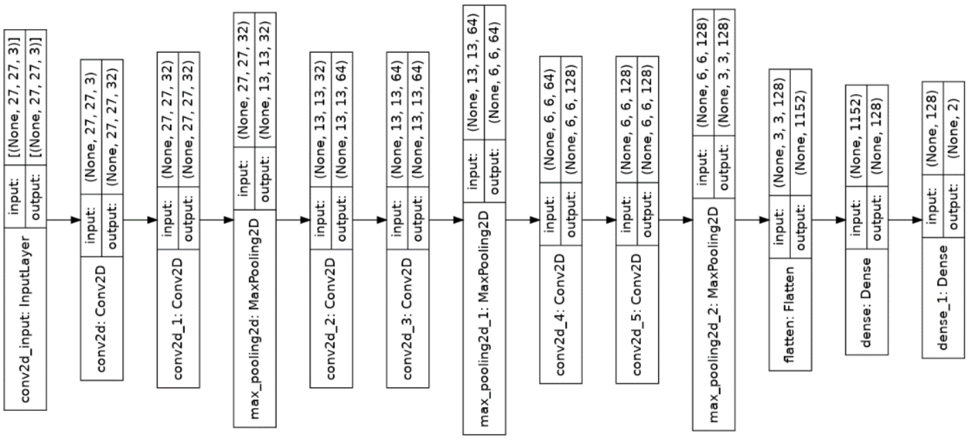
Ultimate Judgement:

After comparing both models we can say that our first model performs really well to predict cancerous or non-cancerous cell, but our second model is also predicting very well and in Deep CNN any accuracy above 70% is assumed to be a good accuracy. So, we can say that both models are suitable for image classification problem.

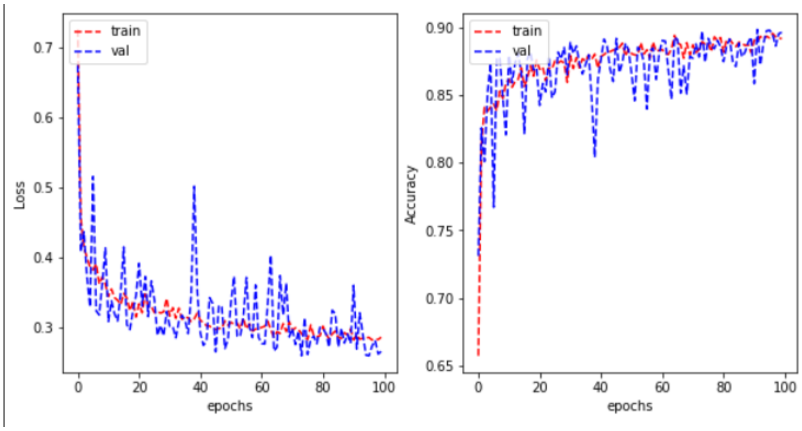
As we discussed in the earlier section for the Task – 3, We've built a model which is a combination of Semi – Supervised Learning and the Transfer Learning. The model doesn't provide better accuracy, because the ratio of unlabelled datasets is higher that the labelled datasets. However, we still can build a better model with the concept of **InPainting**. Using this concept, we'll remove a part of the image and retrain the model in order to predict the part, in this way that the model will learn features of the image by itself, also known as self-supervised learning.

Appendix:

Task – 1:

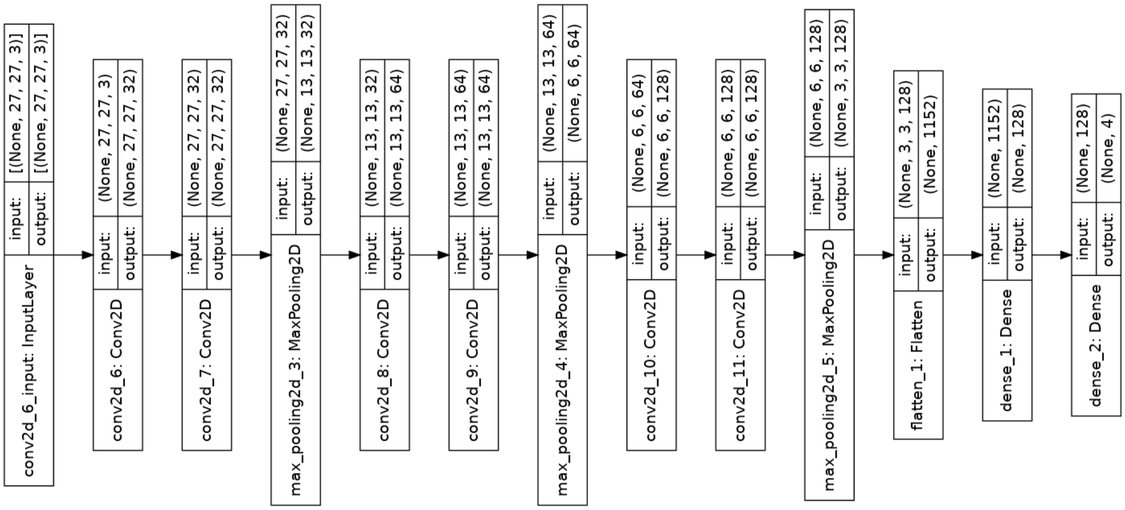

conv2d_input: InputLayer
input: [(None, 27, 27, 3)]
output: [(None, 27, 27, 3)]
conv2d: Conv2D
input: (None, 27, 27, 3)
output: (None, 27, 27, 32)
conv2d_1: Conv2D
input: (None, 27, 27, 32)
output: (None, 27, 27, 32)
max_pooling2d: MaxPooling2D
input: (None, 27, 27, 32)
output: (None, 13, 13, 32)
conv2d_2: Conv2D
input: (None, 13, 13, 32)
output: (None, 13, 13, 64)
conv2d_3: Conv2D
input: (None, 13, 13, 64)
output: (None, 13, 13, 64)
max_pooling2d_1: MaxPooling2D
input: (None, 13, 13, 64)
output: (None, 6, 6, 64)
conv2d_4: Conv2D
input: (None, 6, 6, 64)
output: (None, 6, 6, 128)
conv2d_5: Conv2D
input: (None, 6, 6, 128)
output: (None, 6, 6, 128)
max_pooling2d_2: MaxPooling2D
input: (None, 6, 6, 128)
output: (None, 3, 3, 128)
flatten: Flatten
input: (None, 3, 3, 128)
output: (None, 1152)
dense: Dense
input: (None, 1152)
output: (None, 128)
dense_1: Dense
input: (None, 128)
output: (None, 2)

a. Task – 1 final model

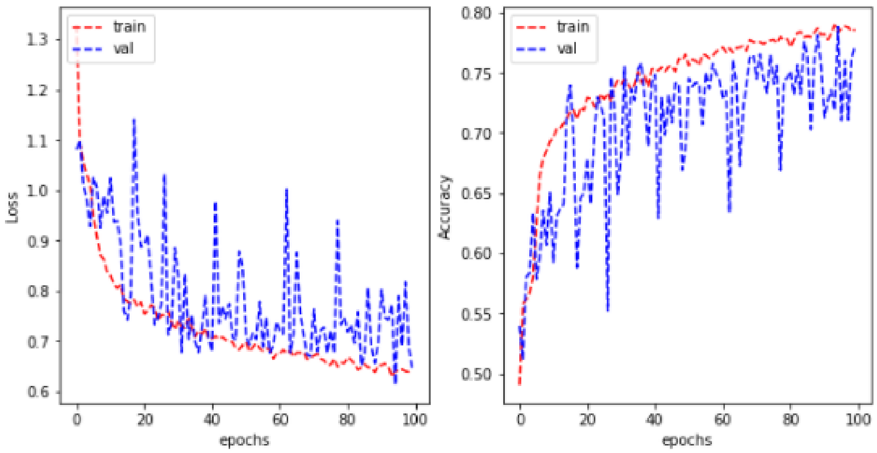


b. Task – 1 Training accuracy and loss

Task – 2:

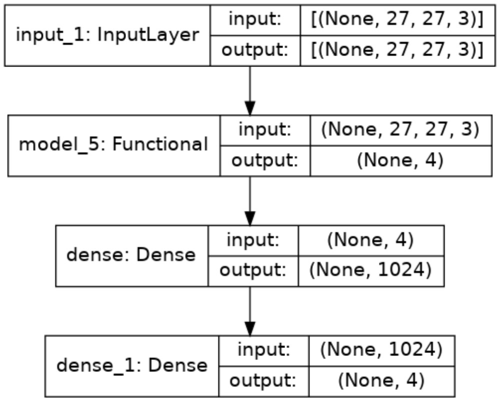

conv2d_6_input: InputLayer
input: [(None, 27, 27, 3)]
output: [(None, 27, 27, 3)]
conv2d_6: Conv2D
input: (None, 27, 27, 3)
output: (None, 27, 27, 32)
conv2d_7: Conv2D
input: (None, 27, 27, 32)
output: (None, 27, 27, 32)
max_pooling2d_3: MaxPooling2D
input: (None, 27, 27, 32)
output: (None, 13, 13, 32)
conv2d_8: Conv2D
input: (None, 13, 13, 32)
output: (None, 13, 13, 64)
conv2d_9: Conv2D
input: (None, 13, 13, 64)
output: (None, 13, 13, 64)
max_pooling2d_4: MaxPooling2D
input: (None, 13, 13, 64)
output: (None, 6, 6, 64)
conv2d_10: Conv2D
input: (None, 6, 6, 64)
output: (None, 6, 6, 128)
conv2d_11: Conv2D
input: (None, 6, 6, 128)
output: (None, 6, 6, 128)
max_pooling2d_5: MaxPooling2D
input: (None, 6, 6, 128)
output: (None, 3, 3, 128)
flatten_1: Flatten
input: (None, 3, 3, 128)
output: (None, 1152)
dense_1: Dense
input: (None, 1152)
output: (None, 128)
dense_2: Dense
input: (None, 128)
output: (None, 4)

c. Task – 2 Final model



d. Task – 1 Training accuracy and loss

Task – 3:

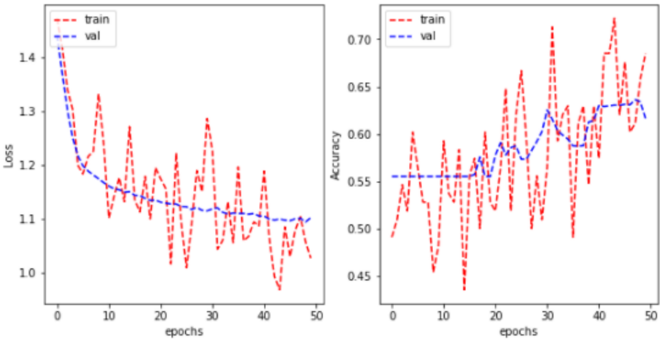


input_1: InputLayer | input: [(None, 27, 27, 3)] | output: [(None, 27, 27, 3)]

model_5: Functional | input: (None, 27, 27, 3) | output: (None, 4)

dense: Dense | input: (None, 4) | output: (None, 1024)

dense_1: Dense | input: (None, 1024) | output: (None, 4)



e. Task – 3 Final batch model

f. Task – 3 final batch training accuracy and loss

References:

1. K. Sirinukunwattana, S. E. A. Raza, Y. Tsang, D. R. J. Snead, I. A. Cree and N. M. Rajpoot, "Locality Sensitive Deep Learning for Detection and Classification of Nuclei in Routine Colon Cancer Histology Images," in IEEE Transactions on Medical Imaging, vol. 35, no. 5, pp. 1196-1206, May 2016, doi: 10.1109/TMI.2016.2525803.
2. Masud, M.; Sikder, N.; Nahid, A.-A.; Bairagi, A.K.; AlZain, M.A. A Machine Learning Approach to Diagnosing Lung and Colon Cancer Using a Deep Learning-Based Classification Framework. Sensors 2021, 21, 748. Available at: <https://doi.org/ 10.3390/s21030748>
3. TensorFlow. 2021. *Convolution Neural Networks* | *TensorFlow Core [online]* Available at : < https://www.tensorflow.org/tutorials/images/cnn>
4. T. Quast, “Utilizing unlabeled data in cell type identification : A semi-supervised learning approach to classification,” Dissertation, 2020 Available at:< http://liu.diva-portal.org/smash/record.jsf?pid=diva2%3A1435226&dswid=-3658>.
5. Simon Jenni, Paolo Favaro, “Self-Supervised Feature Learning by Learning to Spot Artifacts”, Available at: <https://arxiv.org/abs/1806.05024>