Special Topics - Deep Learning(MSAI 495), Winter 2022

Final Report: Lung Cancer Identification

[Team 3: Saumya Singh, Shreyas Lele, Tejul Pandit, Chukwueloka Obi]

Section 1: Project Theme

Lung cancer is the main cause of death related to cancer.[1]. It's a complex disease with numerous clinically significant subtypes. Non-small cell lung cancer accounts for more than 80% of all primary lung cancers (NSCLC). Manual tissue assessment using traditional light microscopy is a reliable method for histological categorization in clinical practice. Due to inter- and intra-tumor heterogeneity, biopsies may not capture the whole disease morphological and phenotypic picture[2]. Furthermore, only 1 or 2 slides are evaluated from each tissue block sent for diagnosis, limiting the pathologist's ability to comprehend and record the full tumor context. Due to a lack of training and knowledge, pathology's old methodologies are difficult to use. The use of a combination of Data Mining tools can aid in the acquisition of mutation patterns in precision oncology.

Given the complexities of lung cancer classification and the limits of present procedures, new clinical data assessment tools are needed to supplement biopsy and help better identify disease features. The current decision-making mechanism is insufficient in assisting medical practitioners in making sound decisions. Recent research has transitioned to deep learning, notably convolutional neural networks (CNNs), in which representative features are automatically learned from data[3]. In many therapeutic settings, this has aided the development of complex multi-parametric algorithms for cognitive decision-making. The development of enhanced systems that can aid in the easy prediction of Lung Cancer using Computer Vision techniques is required.

We use current advancements in deep learning to contribute significantly to improving accuracy and sensitivity in the early stages of cancer in this work. For this task we are working on a Lung Cancer Image Dataset collected from Kaggle[4] comprising 9717 files. This preliminary work demonstrates the potential for deep learning techniques to predict Lung Cancer.

Section 2: Literature Survey

We got motivated to work towards this dataset to glean insights on the Lung Cancer detection methodologies by reading the paper [5]. This paper really explains the need of how necessary it is to deep-dive into this problem. Lung Cancer is the third most common cancer in the United States. In this paper, classifier methods like SVM are used so we thought of taking it a notch higher and implementing Convolution Neural Networks (CNNs) which have proven to be really effective while working with Image Dataset.

On further analysis, we found relevant information in [6]. This research aided us in continuing our current thought process by indicating that the CNN-derived features in this cohort constituted distinct biologic and diagnostic patterns and were linked to underlying tumor microanatomy. This pilot study shows that deep learning-based research has the potential to improve the human-based decision tree for lung cancer classification.

This model was tested on a cohort of 311 patients, and the CNN model was not only helpful in Cancer classification, but it could also distinguish between different stages of Lung Cancer. So, we decided to move ahead with the Deep Learning Model.

Section 3: Dataset Description/ Pre-processing

The choice of dataset for our project is [4] wherein various stages of lung cancer but with different analyses are presented. These stages are distributed in 12 distinct categories that could potentially show indications of lung cancer. The images belong to the stomach, esophagus, and bowels. With the help of the dataset we wish to assess the various sites where lung cancer spreads[7] which can further help to understand the precautionary treatment/ surgery that the patient should undergo.

Three sets of dataset are available for training, validation, and test respectively. The count of total images in each of these sets is as below -

Training set	Validation set	Test set
6958	1956	938

For the first iteration a sample of images from each of the classes are taken, thus making the total count of images for training, validation, and testing as follows -

Training set	Validation set	Test set
5152	1661	964

The distribution of the 12 classes across each set is as shown in Fig. 1-3. The ‘a’ series shows the original count of images available in the 12 different classes. Meanwhile, the ‘b’ series represents the count of images taken post sampling and used for training the model. This helps us understand the variability in count present in the dataset and take the required action during pre-processing.

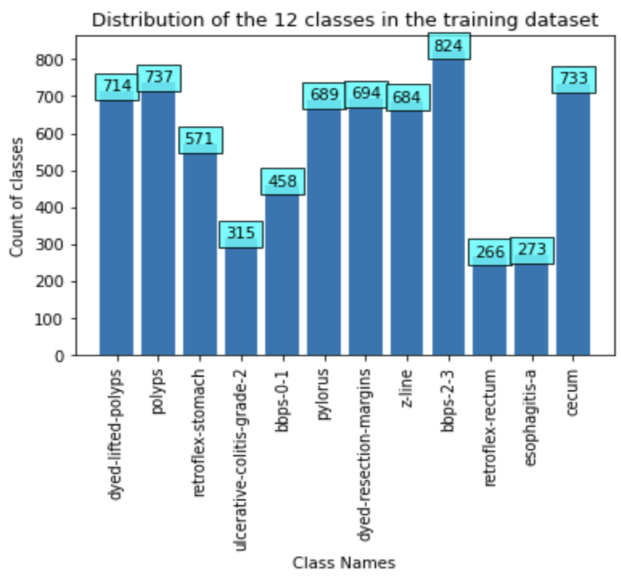


Fig. 1(a): Class distribution in original training dataset

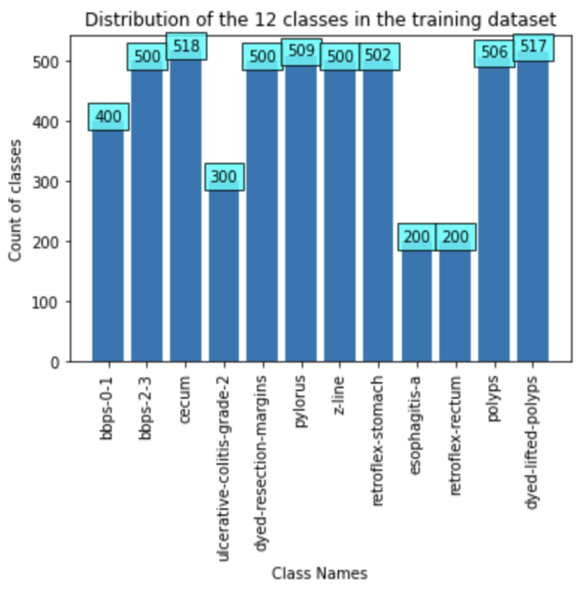


Fig. 1(b): Class distribution in sampled training dataset

Fig. 1(a) provides insight into the disparity of information available across the 12 classes. Since the count of images is not similar across the 12 classes, a subset of images from each class is taken for the initial classification model such that the count of images is similar which is shown in Fig. 1(b). This helps us ensure that none of the classes are overpowering others thus allowing the model to train equally across the various classes.

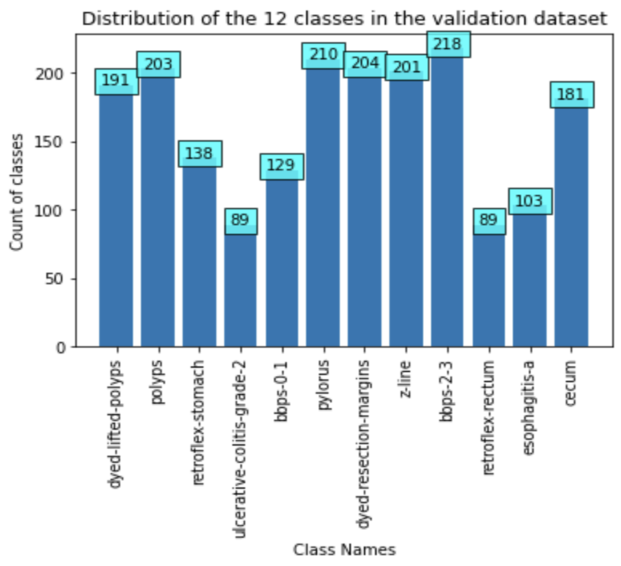


Fig. 2(a): Class distribution in original validation dataset

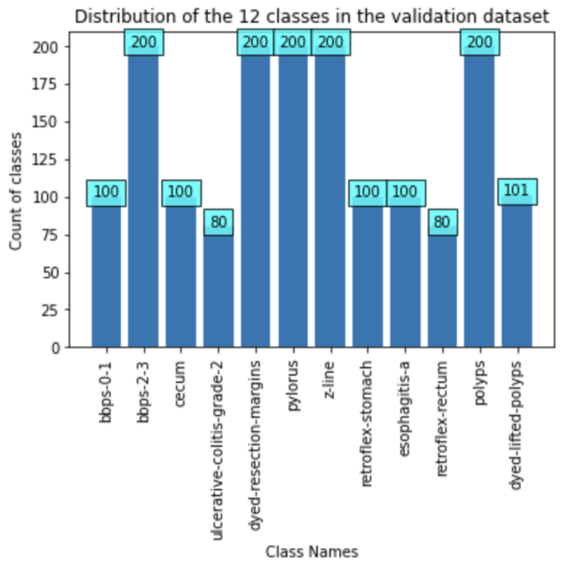


Fig. 2(b): Class distribution in sampled validation dataset

Fig. 2(a) shows the distribution of images in the validation dataset which will be used to assess the model performance at every epoch of model training phase. Similar to training, a subset of images from each class is taken from the validation set as shown in Fig. 2(b). Also, the count of images in each class of validation set is kept less than that of images present in the training sample of the corresponding class.

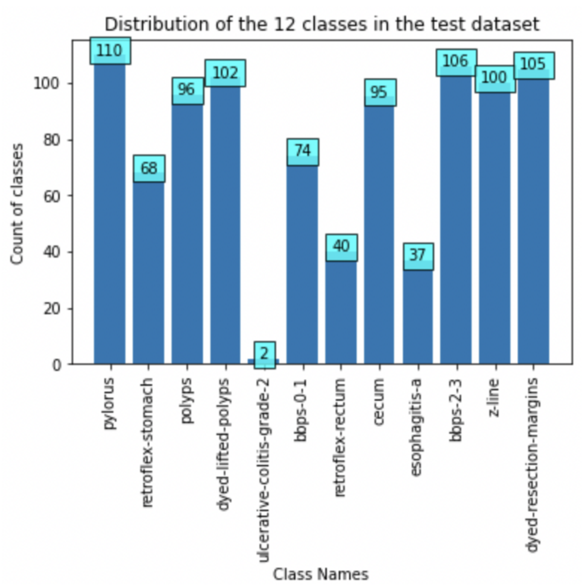


Fig. 3(a): Class distribution in original test dataset

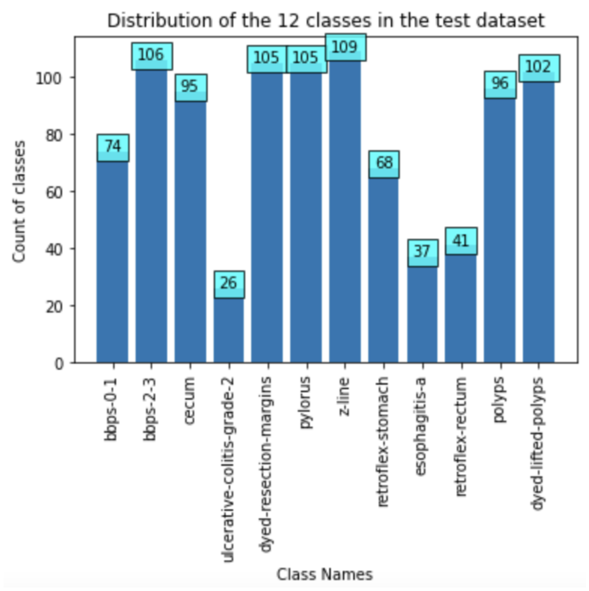


Fig. 3(b): Class distribution in modified test dataset

Lastly, the test dataset will be used to assess the final model performance. The test dataset helps to understand how well the model can translate its learning from training data onto unseen/ test data. Since, one of the classes is empty in the original test dataset as seen in Fig. 3(a), unused images from the training and validation set will be used to fill this gap to create the updated test set shown in Fig. 3(b).

In addition to sampling of images, image pre-processing is added. Tensorflow keras has a functionality called ImageDataGenerator[8] which helps to augment data in real-time by generating batches of tensor images. This helps us to increase the count of images in each set while concurrently providing different views of the same image allowing the model to observe the variations that can be visible. The image generation can include rescaling, zooming, scaling, width/height shift, and horizontal flipping of images. As a result, the model becomes as robust as possible.

Section 4: Prototype

The implementation of our project is inspired by the work done by [9] which targets using deep learning, image processing based, algorithms for detection of lung cancer. We developed a solution that can identify the category of lung cancer the image would belong to from the 12 categories mentioned in Section 3.

As mentioned in Section 1, our project is aimed to implement and improve existing methods to detect lung cancer from images of various organs that are affected by lung cancer. Most implementations in the past were done on x-ray images of lungs or images taken using traditional systems. We utilized data generated from recent technologies in medical science that can capture better information in the form of images. The methodology followed for our initial prototype is as follows -

Data Preparation:

- *Step 1:* As mentioned in Section 2, the images are first sampled to obtain a small subset of images such that the distribution of data is not too varied and the number of images in each class is in a considerable range.
- *Step 2:* Since we decrease the number of images in Step 1, here we augment the images by using the Keras function ImageDataGenerator() as explained in Section 3. The arguments set for our use-case include -
    - rescaling images from 0 to 255 to the scale of 0 to 1 to reduce sparsity present in data and normalizing the data,
    - rotating random images to 40 degrees clockwise to include variations and modifications that can be present in the unseen data,
    - zooming images at random with a range of [0.8, 1.2] which provides the scope of zoom that can take effect on the images,
    - shear random images that can shift a part of image in range [0.8, 1.2] in counter-clockwise direction,
    - randomly shifting images in horizontal or vertical direction.

The generator prepared in Step 2 is applied on the training images folder, the images are resized to 200x200, and split into batches of size 128.

The ImageDataGenerator() for test and validation data rescales the image to [0,1] range and the images in test and validation sets are resized to 200x200, and split into batches of size 128.

Section 5: Model Architecture

Part 1: Convolutional Neural Network with sample of data

A sequential CNN model is built in Keras[10, 13] for classifying images into the 12 categories. The architecture is as shown in Fig. 4. It can be seen that 5 layers of convolution layers are used, each followed by max pooling. The input images are taken as 200x200x3 in the first layer which are then flattened and 12 output neurons are placed in the last layer for the 12 different classes. Also, activation function 'relu' is used throughout the architecture except the last layer. 'Softmax' activation function is used in the last layer to ensure that probabilistic values are assigned to each of the 12 categories. Thus, the class with the highest probability score will be assigned to the unseen image during predictions.

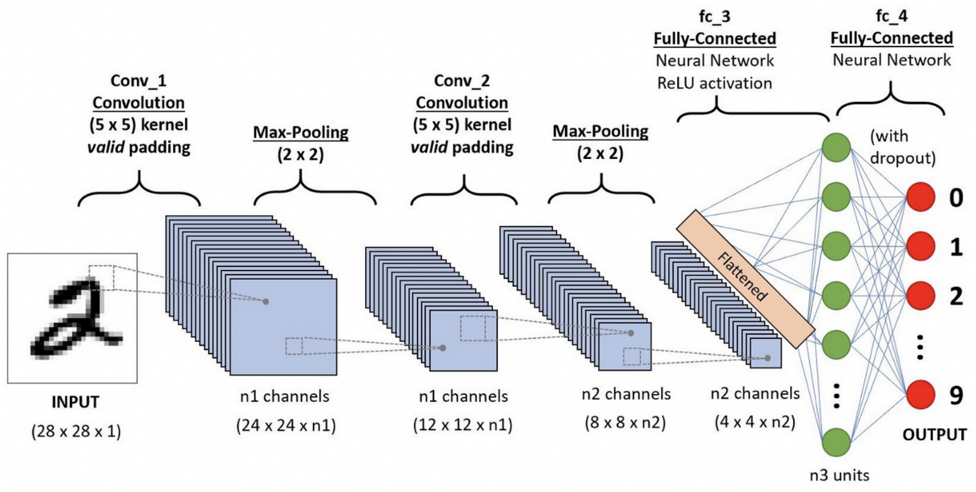


Fig. 4: CNN Model Architecture

Additionally, since the class distribution in the training set is not uniformly distributed, class weights are assigned. The calculation of class weights is done by the compute_class_weight() function in sklearn [11] which is used to estimate class weights for unbalanced datasets. The class weights used in our use-case is as shown in Fig. 5(a).

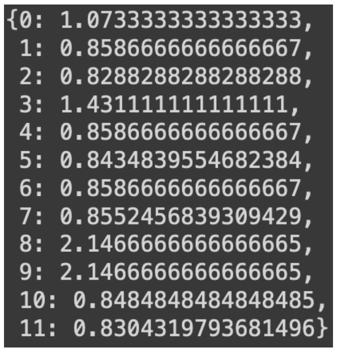

```
{0: 1.0733333333333333,
 1: 0.8586666666666667,
 2: 0.8288288288288288,
 3: 1.4311111111111111,
 4: 0.8586666666666667,
 5: 0.8434839554682384,
 6: 0.8586666666666667,
 7: 0.8552456839309429,
 8: 2.1466666666666665,
 9: 2.1466666666666665,
 10: 0.8484848484848485,
 11: 0.8304319793681496}
```

Fig. 5(a): Class weight distribution

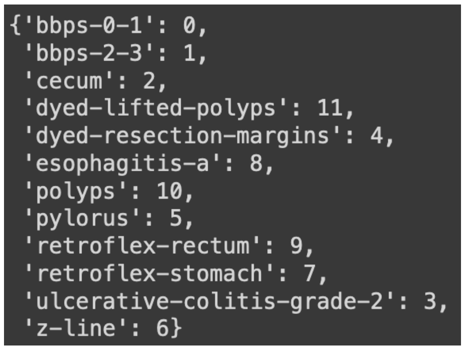

```
{'bbps-0-1': 0,
 'bbps-2-3': 1,
 'cecum': 2,
 'dyed-lifted-polyps': 11,
 'dyed-resection-margins': 4,
 'esophagitis-a': 8,
 'polyps': 10,
 'pylorus': 5,
 'retroflex-rectum': 9,
 'retroflex-stomach': 7,
 'ulcerative-colitis-grade-2': 3,
 'z-line': 6}
```

Fig. 5(b): Class name to numeric map

The keys 0 to 11 in the dictionary shown in Fig. 5(a) signify the 12 classes in the numeric representation. The mapping of each of the classes to its numeric representation is as shown in Fig. 5(b). By passing the class weight dictionary as an argument to the model while fitting on the training set, the loss function uses these weights to assign higher value to classes that have higher weights, signifying that they have less number of samples[12].

The loss function used in the model is categorical crossentropy which compares a one-hot true target with a tensor of predicted targets. Also, the optimizer used in the model is RMSprop with a learning rate of 0.001[13]. The model is trained for 30 epochs with 40 steps for training in each epoch. The number of steps in each epoch is calculated by taking the integer value of the ratio of total number of samples in the training set (5152) to the batch size (128). The training and validation accuracy and loss plot is as shown in Fig. 6.

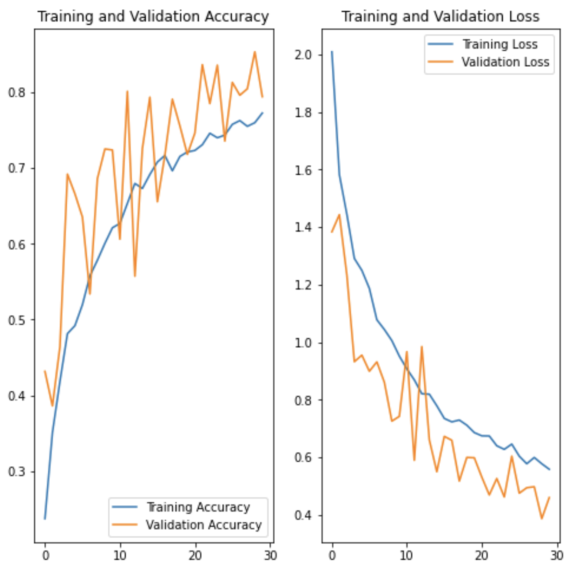
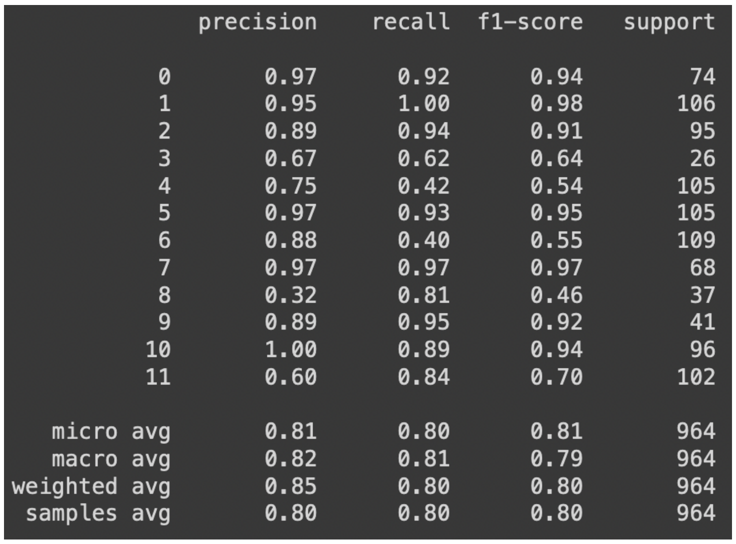


Fig. 6: Training and validation accuracy and loss plot

The final training accuracy is 77.27% and final validation accuracy is 79.43%. It can be seen from Fig. 6 that there is a steady increase in accuracy of training and validation and a corresponding steady decrease in loss of training and validation. Since, training and validation curves are consistently close to each other, it can be concluded that there is no overfitting involved in the training phase.

On further testing the model on unseen samples, the results are shown in Fig. 7. Precision, recall and F-1 scores of each of the classes are calculated when the trained model was used for prediction on images belonging to the test set.



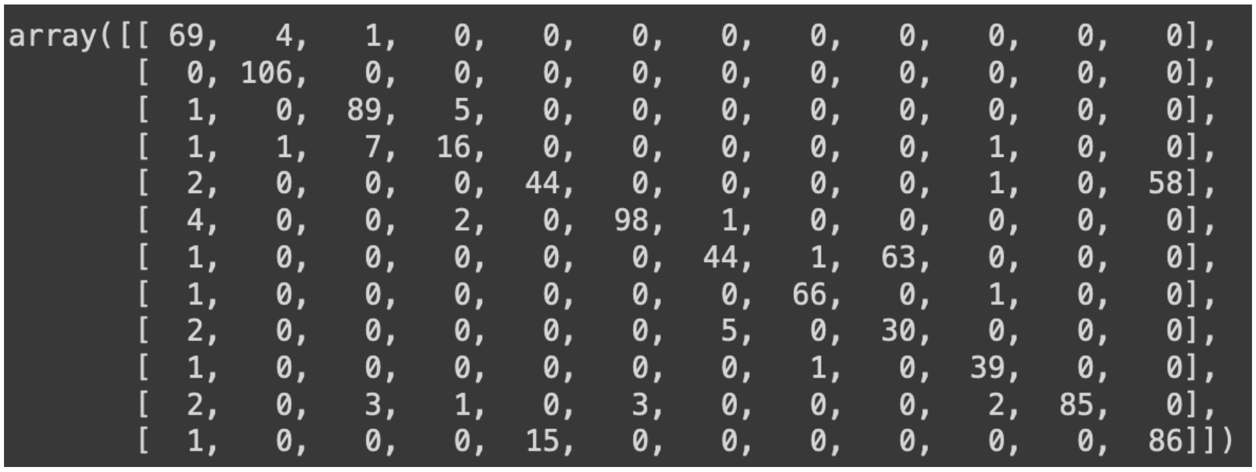
```
              precision    recall  f1-score   support

           0       0.97      0.92      0.94        74
           1       0.95      1.00      0.98       106
           2       0.89      0.94      0.91        95
           3       0.67      0.62      0.64        26
           4       0.75      0.42      0.54       105
           5       0.97      0.93      0.95       105
           6       0.88      0.40      0.55       109
           7       0.97      0.97      0.97        68
           8       0.32      0.81      0.46        37
           9       0.89      0.95      0.92        41
          10       1.00      0.89      0.94        96
          11       0.60      0.84      0.70       102

   micro avg       0.81      0.80      0.81       964
   macro avg       0.82      0.81      0.79       964
weighted avg       0.85      0.80      0.80       964
 samples avg       0.80      0.80      0.80       964
```

Fig. 7: Precision, recall, and F-1 scores on test set

In addition, Fig. 8 provides an insight into the distribution of predictions into each of the classes. The array is a representation of the number or count of images being classified by the model into each of the classes and their actual class (True vs predicted class distribution).



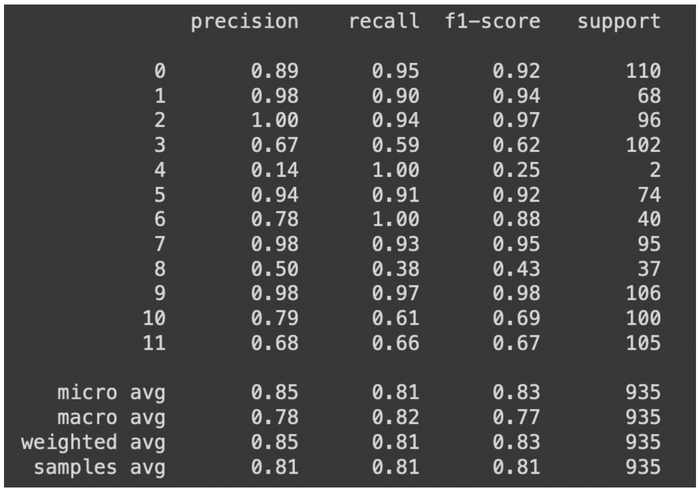
```
array([[ 69,   4,   1,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0, 106,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  1,   0,  89,   5,   0,   0,   0,   0,   0,   0,   0,   0],
       [  1,   1,   7,  16,   0,   0,   0,   0,   0,   1,   0,   0],
       [  2,   0,   0,   0,  44,   0,   0,   0,   0,   1,   0,  58],
       [  4,   0,   0,   2,   0,  98,   1,   0,   0,   0,   0,   0],
       [  1,   0,   0,   0,   0,   0,  44,   1,  63,   0,   0,   0],
       [  1,   0,   0,   0,   0,   0,   0,  66,   0,   1,   0,   0],
       [  2,   0,   0,   0,   0,   0,   5,   0,  30,   0,   0,   0],
       [  1,   0,   0,   0,   0,   0,   0,   1,   0,  39,   0,   0],
       [  2,   0,   3,   1,   0,   3,   0,   0,   0,   2,  85,   0],
       [  1,   0,   0,   0,  15,   0,   0,   0,   0,   0,   0,  86]])
```

Fig. 8: True vs Predicted class distribution

In conclusion, for the first prototype that we have built for the task of classifying images into one of the 12 sections that are affected by lung cancer, we achieve an overall accuracy of 80% on unseen data.

Part 2: Convolutional Neural Network with complete data

To improve on the above model, we then train on the entire dataset with a similar CNN model, except that an additional dense layer is added and the optimizer is changed to Adam keeping the learning rate constant. The overall accuracy of the model on unseen data improved from 80% in the previous attempt to 83%. The precision, recall, and F-1 score distribution across all classes is as shown in Fig. 9 with the corresponding confusion matrix in Fig. 10.

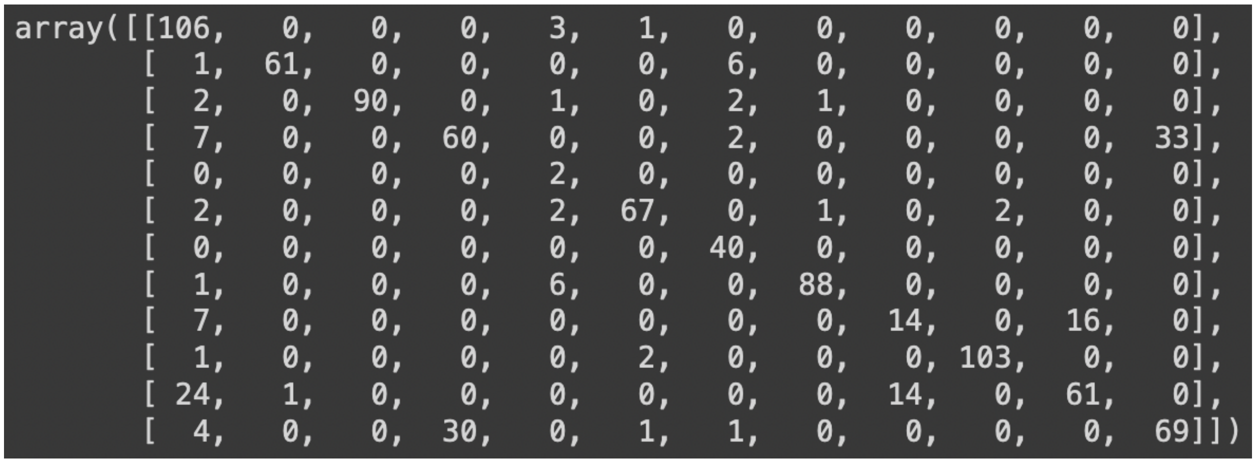


```
              precision    recall  f1-score   support

           0       0.89      0.95      0.92       110
           1       0.98      0.90      0.94        68
           2       1.00      0.94      0.97        96
           3       0.67      0.59      0.62       102
           4       0.14      1.00      0.25         2
           5       0.94      0.91      0.92        74
           6       0.78      1.00      0.88        40
           7       0.98      0.93      0.95        95
           8       0.50      0.38      0.43        37
           9       0.98      0.97      0.98       106
          10       0.79      0.61      0.69       100
          11       0.68      0.66      0.67       105

   micro avg       0.85      0.81      0.83       935
   macro avg       0.78      0.82      0.77       935
weighted avg       0.85      0.81      0.83       935
 samples avg       0.81      0.81      0.81       935
```

Fig. 9: Precision, recall, and F-1 scores on test set



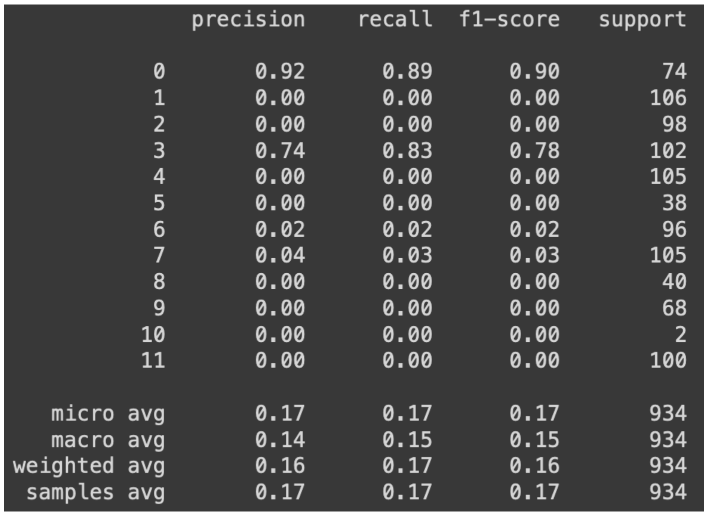
```
array([[106,   0,   0,   0,   3,   1,   0,   0,   0,   0,   0,   0],
       [  1,  61,   0,   0,   0,   0,   6,   0,   0,   0,   0,   0],
       [  2,   0,  90,   0,   1,   0,   2,   1,   0,   0,   0,   0],
       [  7,   0,   0,  60,   0,   0,   2,   0,   0,   0,   0,  33],
       [  0,   0,   0,   0,   2,   0,   0,   0,   0,   0,   0,   0],
       [  2,   0,   0,   0,   2,  67,   0,   1,   0,   2,   0,   0],
       [  0,   0,   0,   0,   0,   0,  40,   0,   0,   0,   0,   0],
       [  1,   0,   0,   0,   6,   0,   0,  88,   0,   0,   0,   0],
       [  7,   0,   0,   0,   0,   0,   0,   0,  14,   0,  16,   0],
       [  1,   0,   0,   0,   0,   2,   0,   0,   0, 103,   0,   0],
       [ 24,   1,   0,   0,   0,   0,   0,   0,  14,   0,  61,   0],
       [  4,   0,   0,  30,   0,   1,   1,   0,   0,   0,   0,  69]])
```

Fig. 10: True vs Predicted class distribution

Part 3: Using pre-trained model (Resnet-152) on complete data

We additionally wanted to improve on the model by leveraging the large pre-trained models. We wanted to compare the performance of our model built for our dataset with large 50+ layered CNN models trained for another task that can be fine-tuned for our images. Transfer learning is an approach where we can utilize the bottleneck/ penultimate and last layers for our individual classification task[14]. The models we plan to use are Resnet[15]. Resnet is a simple but effective architecture introduced in 2015 and one of the variants has 152 layers and was popular for its effective way of working with vanishing gradients when more layers are introduced.

The overall accuracy of the Resnet model on validation data was 85%, which was a significant improvement considering that we trained the bottleneck layers for only 4 epochs. However, the model could not translate the same output to the test dataset due to the limited number of epochs the model was trained on. The precision, recall, and F-1 score distribution across all classes in the test set is as shown in Fig. 11 with the corresponding confusion matrix in Fig. 12.

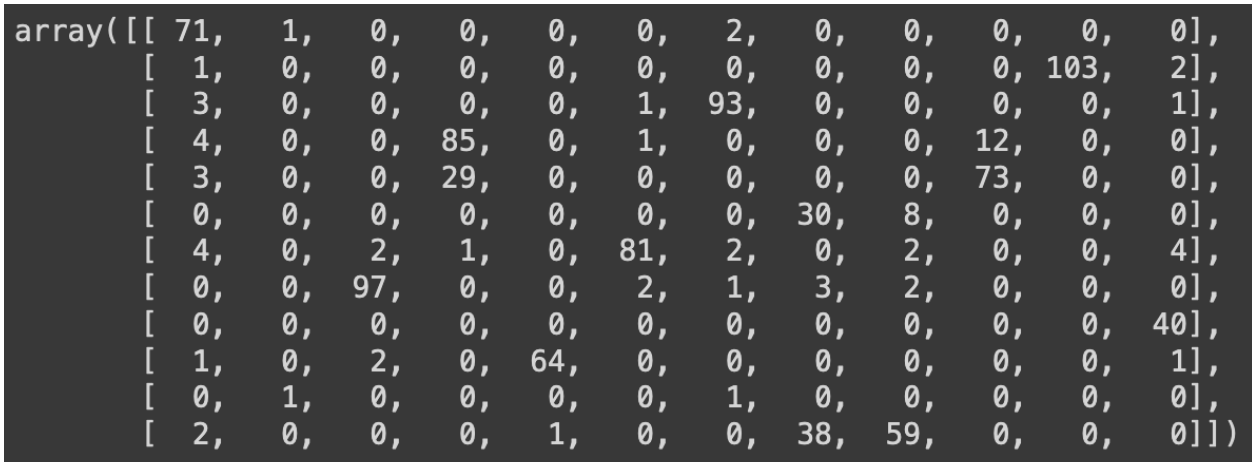


```
              precision    recall  f1-score   support

           0       0.92      0.89      0.90        74
           1       0.00      0.00      0.00       106
           2       0.00      0.00      0.00        98
           3       0.74      0.83      0.78       102
           4       0.00      0.00      0.00       105
           5       0.00      0.00      0.00        38
           6       0.02      0.02      0.02        96
           7       0.04      0.03      0.03       105
           8       0.00      0.00      0.00        40
           9       0.00      0.00      0.00        68
          10       0.00      0.00      0.00         2
          11       0.00      0.00      0.00       100

   micro avg       0.17      0.17      0.17       934
   macro avg       0.14      0.15      0.15       934
weighted avg       0.16      0.17      0.16       934
 samples avg       0.17      0.17      0.17       934
```

Fig. 11: Precision, recall, and F-1 scores on test set



```
array([[ 71,   1,   0,   0,   0,   0,   2,   0,   0,   0,   0,   0],
       [  1,   0,   0,   0,   0,   0,   0,   0,   0,   0, 103,   2],
       [  3,   0,   0,   0,   0,   1,  93,   0,   0,   0,   0,   1],
       [  4,   0,   0,  85,   0,   1,   0,   0,   0,  12,   0,   0],
       [  3,   0,   0,  29,   0,   0,   0,   0,   0,  73,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  30,   8,   0,   0,   0],
       [  4,   0,   2,   1,   0,  81,   2,   0,   2,   0,   0,   4],
       [  0,   0,  97,   0,   0,   2,   1,   3,   2,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  40],
       [  1,   0,   2,   0,  64,   0,   0,   0,   0,   0,   0,   1],
       [  0,   1,   0,   0,   0,   0,   1,   0,   0,   0,   0,   0],
       [  2,   0,   0,   0,   1,   0,   0,  38,  59,   0,   0,   0]])
```

Fig. 12: True vs Predicted class distribution

Section 6: Conclusion

The project helped us achieve a solution to identify the potential threats that an individual may be affected by due to lung cancer. We believe that using a system like ours in conjunction with techniques used by medical practitioners could potentially help them assess the patient in an improved setting. Additionally, we examine from our exercise that pre-trained algorithms provide an overall much better performance as compared to traditional architectures despite being trained for less number of epochs and trained over a completely different dataset over the final layers only.

References

[1] Huang, T. et al. Distinguishing lung adenocarcinoma from lung squamous cell carcinoma by two hypomethylated and three hypermethylated genes: a meta-analysis. PLoS ONE 11, e0149088 (2016).

[2] Ilié, M. & Hofman, P. Pros: Can tissue biopsy be replaced by liquid biopsy?. Transl. Lung Cancer Res. 5, 420–423 (2016).

[3] Hosny, A. et al. Deep learning for lung cancer prognostication: a retrospective multi-cohort radiomics study. PLoS Med 15, e1002711 (2018).

[4] Balasubramaniam (2022, January). lung_cancer_dataset, Version 1. Retrieved February 4, 2022 from https://www.kaggle.com/balasubramaniamv/lung-cancer-dataset/version/1.

[5] Aliferis, Constantin & Tsamardinos, Ioannis & Massion, Pierre & Statnikov, Alexander & Fananapazir, Nafeh & Hardin, D.. (2003). Machine Learning Models For Classification Of Lung Cancer and Selection of Genomic Markers Using Array Gene Expression Data. 67-71.

[6] Chaunzwa, T.L., Hosny, A., Xu, Y. *et al.* Deep learning classification of lung cancer histology using CT images. *Sci Rep* **11,** 5471 (2021). https://doi.org/10.1038/s41598-021-84630-x.

[7] Lynne Eldridge, MD. “Where Does Lung Cancer Spread and How Can I Know If It Has?” *Verywell Health*, https://www.verywellhealth.com/where-does-lung-cancer-spread-2249368.

[8] “Tf.keras.preprocessing.image.imagedatagenerator : Tensorflow Core v2.8.0.” *TensorFlow*, https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator.

[9] R. Tekade and K. Rajeswari, "Lung Cancer Detection and Classification Using Deep Learning," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1-5, doi: 10.1109/ICCUBEA.2018.8697352.

[10] Chollet, F. (2016, June 5). *Building powerful image classification models using very little data*. The Keras Blog. Retrieved February 28, 2022, from https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

[11] *sklearn.utils.class_weight.compute_class_weight — scikit-learn 1.0.2 documentation*. (n.d.). Scikit-learn. Retrieved February 28, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html

[12] *How to set class weights for imbalanced classes in Keras?* (n.d.). Data Science Stack Exchange. Retrieved February 28, 2022, from https://datascience.stackexchange.com/questions/13490/how-to-set-class-weights-for-imbalanced-classes-in-keras

[13] *Keras API reference*. (n.d.). Keras. Retrieved February 28, 2022, from https://keras.io/api/

[14] Brownlee, Jason. “A Gentle Introduction to Transfer Learning for Deep Learning.” *Machine Learning Mastery*, 16 Sept. 2019, https://machinelearningmastery.com/transfer-learning-for-deep-learning/.

[15] *[1512.03385] Deep Residual Learning for Image Recognition*. (2015, December 10). arXiv. Retrieved February 28, 2022, from https://arxiv.org/abs/1512.03385

[16] Tan, M., & ., Q. V. (n.d.). *[1905.11946] EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. arXiv. Retrieved February 28, 2022, from https://arxiv.org/abs/1905.11946

[17] Adaloglou, N. (2021, January 21). *Best deep CNN architectures and their principles: from AlexNet to EfficientNet*. AI Summer. Retrieved February 28, 2022, from https://theaisummer.com/cnn-architectures/#resnet-deep-residual-learning-for-image-recognition-2015