Classifying and Localizing Metastatic Tumor Cells Using Deep Neural Networks

Team #2 Final Report

Yufei Lin

Data Science, Worcester Polytechnic Institute, Worcester, MA, ylin9@wpi.edu

Ashley Schuliger

Computer Science, Worcester Polytechnic Institute, Worcester, MA, amschuliger@wpi.edu

Parker Simpson

Bioinformatics and Computational Biology, Worcester Polytechnic Institute, Worcester, MA, pjsimpson@wpi.edu

ABSTRACT

The lymphatic system is an integral part of our immune system; therefore, efficient identification of tumor cells in this system is an important task for health care professionals. Deep neural networks are a powerful tool for image classifications and object localization, making their application in the medical field, with scans and microscope images, an exciting new field of research. Here, we have proposed two models, CNN-RNN and transfer learning based on ResNet50, for classification of images, and used explainable AI (XAI) for tumor cell identification.

KEYWORDS

RNN, CNN, Transfer Learning, ResNet50, Classification, Localization, Test-time Augmentation, Explainable AI

1 Introduction

The lymphatic system is a network of vessels, tissues, and organs that help remove waste and toxins from the body. Lymph, a watery fluid containing white blood cells, is transported throughout the body using lymph vessels and structures called lymph nodes. Although lymph fluid contains some white blood cells, the majority of protection against germs and infection by the lymphatic system occurs at the lymph nodes. Because lymph nodes filter out toxins and unhealthy cells, swelling in a node can indicate that something unusual is occurring near the nodes, like a fever, or in rarer cases cancer. Not all enlarged lymph nodes contain cancer, in fact most do not, but the only way to definitively determine if a node contains cancer is through biopsy. The biopsied lymph node tissue sample is examined under a microscope for the presence of cancerous cells. This examination process is currently carried out by a physician, but advances in computational research have enabled us to change that.

After aggregating many histopathological microscope images of lymph node sections from biopsies along with labels from humans that indicate whether or not the image contains cancerous cells, the dataset called PatchCamelyon (PCam) has been created. The goal of our project, and the purpose for which the PCam dataset was created, is to create an algorithm to identify the presence of cancer cells within small image patches from larger histopathological microscope images. Algorithms for this kind of diagnosis are becoming increasingly more useful as the amount of data increases, the power of computing advances, the healthcare systems step further into the digital age. An algorithm that can diagnose scans on at least the same level as a human can expedite the care process for patients by decreasing the amount of human interaction for a scan before action is taken. This type of task deals with the tasks of image classification and object localization. Since the goal of our project is to classify images based on whether tumor cells are present or not, we will perform image classification. However, in order to do this, we must be able to identify locations in the image that contain tumors in order to accurately make this classification. In the subsequent sections, we propose two deep neural network structures for identifying tumor cells in small images patches. We discuss the methodology used for building our CNN-RNN and ResNet50 neural networks as well as the training and validation process that we used in order to obtain our models. We then train our model using cross validation to measure the accuracy. We also experiment with adjusting our hyperparameters in order to increase our accuracy further. Once trained, we test our complete neural networks on a held out dataset and achieve accuracies of 82% and 80% respectively. We also provide a methodology for interpreting our model using explainable AI.

2 Data

The Kaggle contest we took on uses a slightly abridged version of the PCam dataset that contains no duplicate images while maintaining the same data split as the original dataset. This dataset will be referred to as kPCam going forward. The kPCam dataset consists of 96x96 pixel color images These images are labeled with either a positive or negative label. A positive label indicates that the 32x32 pixel bounding box in the given image contains at least one pixel that contains a tumor cell. A negative label indicates that the bounding box on the given image does not contain any pixels with tumor cells. The output of our prediction model will be 0 or 1, where 0 indicates that the image does not contain any tumor cells, and a 1 indicates the presence of tumor cells.

3 Methodology

In this section, we provide a set of methodologies by which to accurately predict and localize the presence of tumor cells in a patient's tissue. We also describe the data processing necessary for building a robust neural network for prediction.

3.1 Data Preprocessing

As mentioned above each image needed to be cropped from its original size of 96x96 pixels to the center 32x32 square before use in the model. Our dataset consists of approximately 220,025 images however, in order to achieve a manageable training duration and memory requirement, we chose to restrict the number of images trained on the model at one time to 4,000 images. Furthemore, in terms of the data input for testing, our model provides an implementation of Test-time Augmentation (TTA). Test time augmentation is a technique where we make different transformations on the input x, for instance horizontal transformation or vertical transformation, and gather output calculated out of all different x and find the dominant class [8]. In our project, we have provided test-time augmentation on multiple different models to help evaluate whether the technique is helpful to construct our final prediction. We will be introducing the influence it has on the accuracy of the two models we have created.

3.2 Models

3.2.1 CNN-RNN

As mentioned in subsequent sections, our preprocessed data consists of 96x96 pixel images with a dimension for each color. The goal of our project is to find tumor cells within images, so it was important for us to choose a model that could accurately extract information using the temporal locality between pixels as well as analyze the relationship between such pixels in order to localize objects (in our case, tumor cells). Thus, we chose to build a neural network that consists of three convolutional layers for extracting information from the image as a whole and a final recurrent neural network (RNN) layer that analyzes the relationship between sequences of pixels. We then fed those outputs into a linear layer and applied the sigmoid activation function to reduce the dimension to two values between 0 and 1, indicating the probability that the image contains tumors or does not. Our final CNN-RNN structure used output sizes of 12, 24, and 48 for each of the respective convolutional layers and an output dimension equal to the input dimension of 12 for the RNN layer. The final linear layer contained 2 output dimensions as previously mentioned. We accounted for the colors of the images by using an input dimension of 3. This structure also utilized batch normalization for optimization and Relu as an activation function between each pair of layers. We also utilized dropouts for optimization, where the dropout percentages are 20%, 10%, and 0% after the convolutional layers respectively. We will discuss how these hyperparameters were chosen in subsequent sections. We also built two simple convolutional neural networks (CNN) to classify each image, but as we will see in subsequent sections, our CNN-RNN structure far outperformed these simple networks.

3.2.2 Transfer Learning - ResNet50

Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned [4]. It is proposed to overcome the problem that many data mining and machine learning algorithms are taking a long time to train; therefore, using some pre-existing models with similar data features would help to combat that problem [5]. This is also a widely adapted technique in image classification because there are many existing models having similar prototypes for different images and shifting them for new purposes is usually easier than training a new model [6]. In our model, we choose to apply ResNet50 as the basis model. ResNet50 model provides 50 layers of ResNet and has relatively low error on top-1 and top-5 layers [7]. Also, this model is built on CNN, which is the most crucial network for image classification and the ones we are using for all of our other models. Given that ResNet input is an image of size 224x224, we have resized our image using pytorch transformation and made normalization of the images using the mean and standard deviation from the following table. We only have a binary classification in the question instead of several classes, where ResNet was first created for, we have added in a new linear layer of the dimension 2048x1 and using Sigmoid activation function to make the prediction.

We have also written two functions to change the dimension for the image to be able to fit in the ResNet convolutional network layer and max pooling layer. The choice of our loss function is binary cross entropy loss and the optimizer was Adam.

Table 1: Hyperparameter for Transfer Learning

Variable name	Value
Mean	[0.485, 0.456, 0.406]
Standard Deviation	[0.229, 0.224, 0.225]

Batch Size	64
Num Epochs	10

3.3 Explainable AI (XAI)

As our classification models stand, there is no way to interpret or understand why our model predicted the presence of tumor cells in an image. To perform this task of localizing tumor cells, we used LIME, which is an explainable AI technique that interprets the classification produced by a black box model. This technique is crucial to solving our problem as it provides medical professionals and researchers with more information surrounding which regions of tissues are important in identifying the presence of tumor cells. Although we are not medical professionals ourselves, we use LIME in subsequent sections to identify regions that our models use for predicting the presence of tumor cells.

3.4 Training and Validation Process

When training our neural networks, we used a combination of loss measurements and gradient descent techniques to obtain the optimal model. We use Cross Entropy Loss to determine the performance of the given model, and then we perform backwards propagation to update our weights accordingly. From here, we step forward once again using our gradient descent mechanism with the given learning rate. For gradient descent, we chose to use Pytorch's Adam gradient descent function. This version of gradient descent obtains an optimal model by performing the normal RMSProp gradient descent with momentum taken into account. This form of gradient descent performs well when there are multiple local minimum losses, as it allows for more exploration to find the global minimum. We regulate the number of times that gradient descent is performed and the weights of the neural networks are updated using a for loop with a number of epochs. In subsequent sections, we discuss how we chose the number of epochs and the learning rate for our gradient descent function.

4 Evaluation and Results

This section will cover the performance of our above mentioned neural network structures. We will first discuss the training and validation results using 5-fold cross validation of the original data set. Next, we evaluate the performance of our models against baseline neural networks. We also provide an explanation for how we chose the correct hyperparameters for our resulting neural networks. Once we have justified our models, we will discuss the results that our explainable AI model produced in order to better interpret the classifications of our models.

4.1 Classification

In the following section, we have provided a detailed explanation of the training and testing results from both of the two classification models mentioned above.

4.1.1 CNN-RNN

4.1.1.1 Results

We start by providing the results of training our CNN-RNN model in order to justify our chosen network structure. Our final model was trained using three convolutional layers with output sizes of 12, 24, and 48 respectively, a RNN layer with an output dimension of 12, and a simple linear layer. These outputs were fed into a sigmoid activation function in order to obtain the final results. In regards to training, we trained our model with 30 epochs and a learning rate of 0.0001. Initially, we split our data into a training set and a testing set, but we wanted to try out different folds of our dataset. Thus, we performed 5-fold cross validation to provide us with a wider range of models to choose from. Figure 1 below shows the resulting training and testing accuracies after performing 5-fold cross validation.

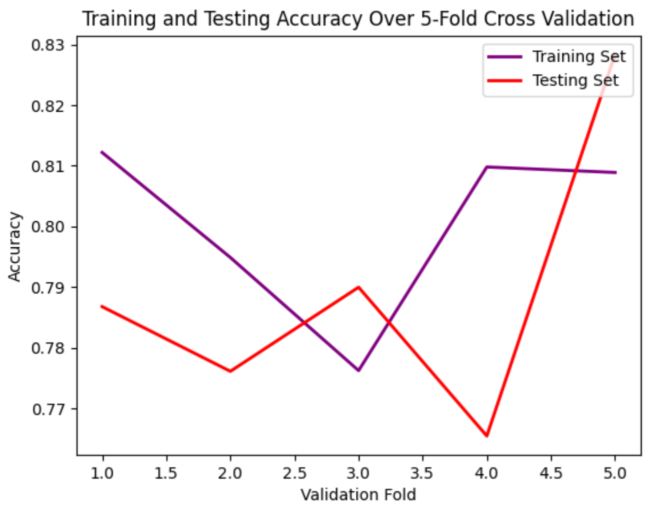

Figure 1: Training and testing accuracies obtained through training our CNN-RNN network

As shown in the figure above, our highest accuracy for both training and testing was achieved by our fifth validation fold with a training accuracy of 80.89% and a testing accuracy of 82.84%. These results provide proof that this network structure provides a strong prediction for the presence of tumor cells in images of tissue. In order to fully validate the performance of this model, we held out approximately 3,000 images to perform a final validation. Using this optimal model, we achieved a 81.32% accuracy on this held out data, which implies a strong performance of our model.

4.1.1.2 **Hyperparameters**

Once we had the structure for our CNN-RNN neural network, we worked towards increasing our accuracy by adjusting the hyperparameters used in our training process. The hyperparameters that we chose to tune include the learning rate for gradient descent, the percentage of dropout used between convolutional layers, and the usage of a LeakyRelu activation function or the Relu activation function. Along with this, we attempted to increase our accuracy by adding another simple linear activation function to our neural network. For the learning rate, we wanted to find the optimal rate for our Adam optimizer that would allow our model to consistently learn rather than jump from a high accuracy to a low accuracy due to a large learning rate. We started with a value of 0.01 and experimented with values of 0.0001, 0.001, 0.01, 0.1, and 1 in order to find the optimal rate. Figure 2 below shows the performance of our CNN-RNN with varying learning rates and all other hyperparameters held constant.

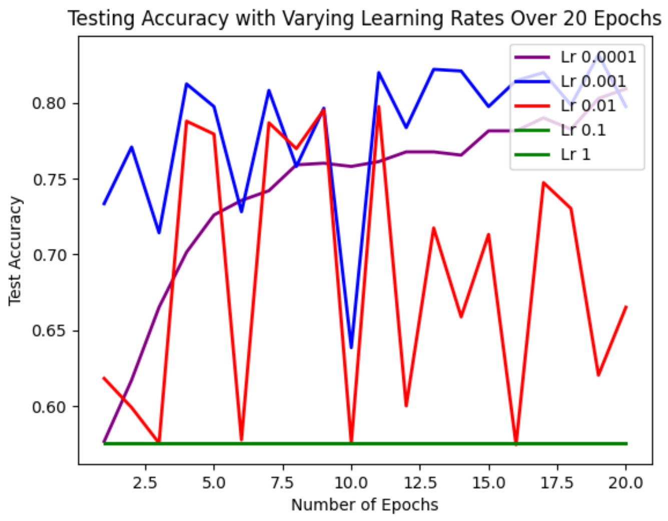

Figure 2: Testing accuracies for various learning rates across 20 epochs using our CNN-RNN network

As shown in the figure above, as we increased this value, the accuracy hovered at approximately 50%, indicating that the model was unable to train. As the learning rate decreased, the model began to train with the accuracy jumping from 57% to 75% over one epoch. As we continued to decrease the learning rate, we found our optimal rate at 0.0001. This learning rate allowed our model to gradually learn and achieve an optimal accuracy of 80.92%. Thus, our final model uses a learning rate of 0.0001 for our optimizer. For dropout, we experimented with a variety of different dropout percentages between the convolutional layers. Dropout is a regularization technique that often helps with models that tend towards overfitting. In theory, the dropout rates between layers could vary significantly depending on the model, as they could all differ or use identical percentages. Due to this, it was important for us to test out a wide range and combination of dropout percentages. In order to do this, we experimented with dropout percentages of 0%, 10%, 20%, and 50%, where each layer used the same percentage. We also tested out the following combinations of percentages for the layers: 50%, 20%, and 0%; 50%, 20%, and 10%; 50%, 50%, and 20%; 20%, 10%, and 0%; 20%, 20%, and 10%; 10%, 10%, and 0%. Figures 3 and 4 below shows the resulting testing accuracies for each combination of dropout percentages.

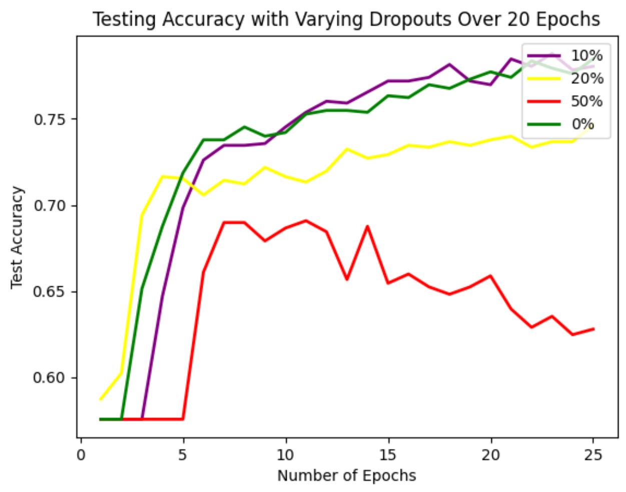

Figure 3: Testing accuracies for various dropout percentages using our CNN-RNN network.

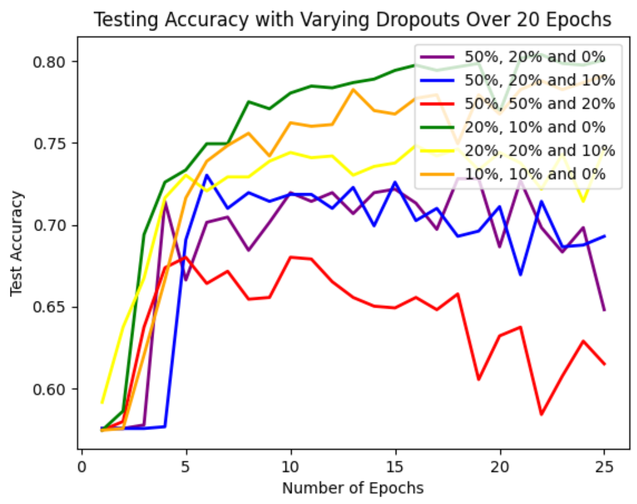

Figure 4: Testing accuracies for various dropout percentage combinations using our CNN-RNN network.

Based on the above results, the highest testing accuracy can be achieved using a combination of 20%, 10%, and 0%. All of our combinations that include 50% appear to perform poorly most likely due to the fact that 50% dropout excludes too many neurons from the training model. We also notice that in general, the combinations of decreasing dropout percentages tend to perform better than the constant dropout percentages. This is logical since the later layers in the network work to extract more specific information regarding the image while the first few layers perform general image feature extractions. As a result, we chose to use a combination of 20%, 10, and 0% for our final model. For our Relu activation function, we experimented with using Relu or LeakyRelu between our convolutional layers in order to increase our accuracy. Figure 5 below shows the resulting testing accuracies of using LeakyRelu versus Relu in our model.

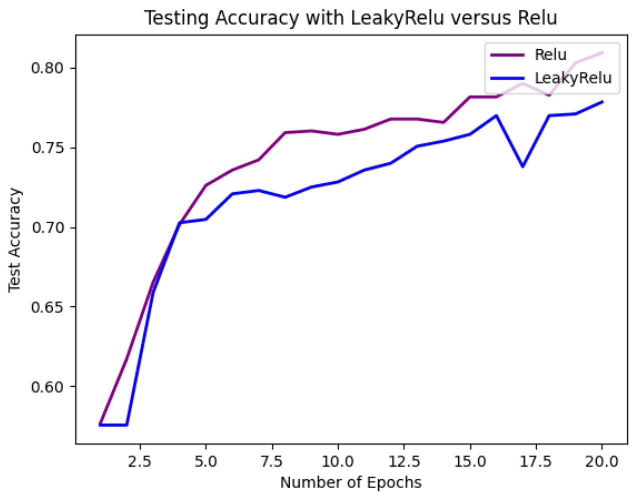

Figure 5: Testing accuracies for our CNN-RNN network using the Relu versus LeakyRelu activation function.

Based on the above figure, the CNN-RNN model using the Relu activation function outperforms the model using LeakyRelu. Due to this, our optimal model will use Relu as an activation function between each pair of convolutional layers. As a result of the above tests, we chose to adjust our model to use a learning rate of 0.0001, dropout percentages of 20%, 10%, and 0%, and the Relu activation function.

4.1.2 **Transfer Learning - ResNet50**

After 10 epochs of training, our Resnet has been able to converge on the training loss, shown in the following figure.

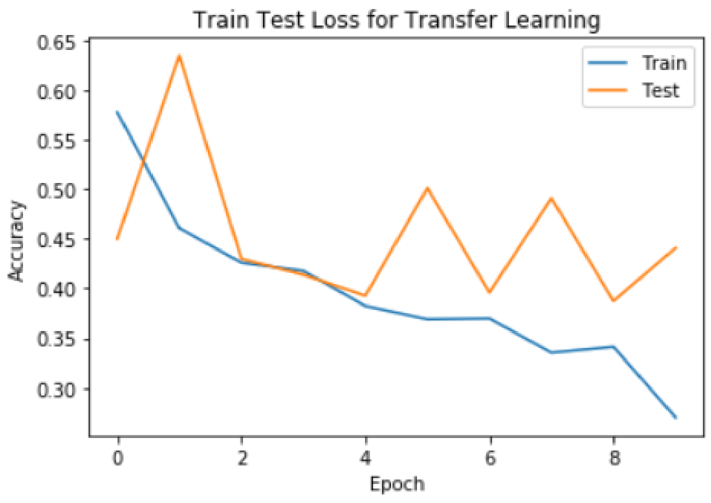

Figure 6: Training and Testing Loss for Transfer Learning

We have our accuracy for training and testing shown as the following. We could see that the accuracy converges after 10 times of training.

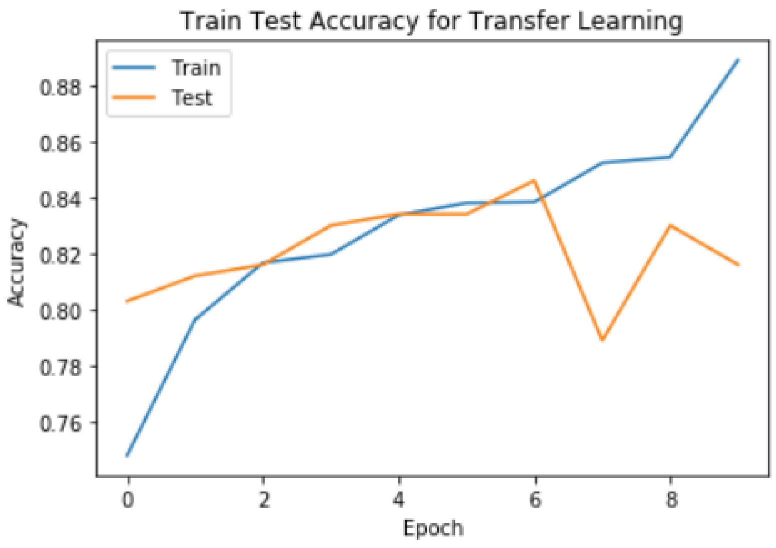

Figure 7: Training and Testing Accuracy for Transfer Learning

We are able to obtain 80% accuracy on the testing set from the trained model. However, after we applied test-time augmentation at the testing stage for both transfer learning and a simple CNN network, no big difference was given on the accuracy. Thus, we cannot conclude that test-time augmentation is effective in making the process more accurate.

4.1.3 **Performance Compared to Baseline**

Based on the results above, both our CNN-RNN and ResNet networks provide a strong prediction for the presence of tumor cells in a patch of tissue. In order to provide further proof of this, we compared our final results to the results produced by two baseline models. Our baseline models are both simple convolutional neural networks with a final linear layer. The output sizes of these baseline models differ, with one using output sizes of 32, 64, and 128 and the other using output sizes of 12, 24, and 48. Similarly to our final models, these baseline models applied the sigmoid activation function to the final output in order to produce our output in terms of 0 and 1. We performed 5-fold cross-validation on these baseline models to obtain classification results similar to those obtained in the previous section. Figure # below shows the training and test accuracies of both baseline models across 5-fold.

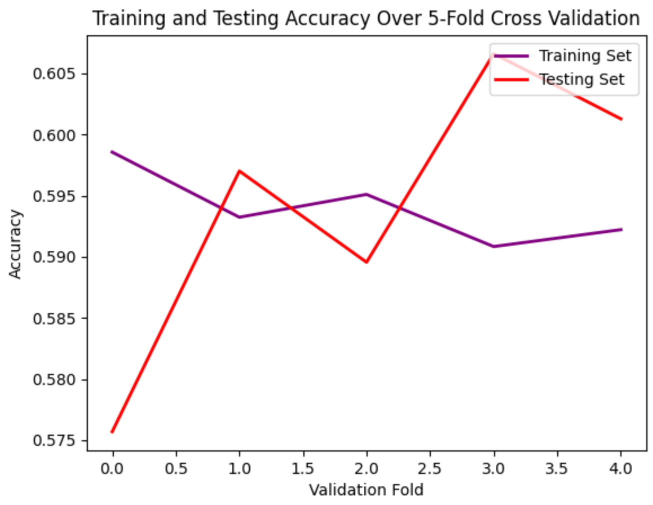

Figure 8: Training and testing accuracies for the simple convolutional neural network baseline with higher output sizes without the use of an RNN layer.

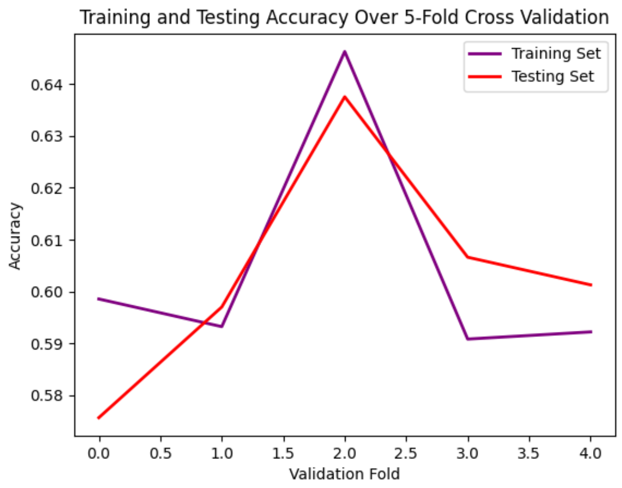

Figure 9: Training and testing accuracies for the simple convolutional neural network baseline with lower output sizes without the use of an RNN layer.

The figure above shows that our baseline models perform only slightly better than random classification, as the high testing accuracy obtained by our baseline models was 60.66%. This weak performance by our baseline models justifies the need for a network that can extract information related to the localization of tumor cells. Thus, our previous CNN-RNN and ResNet models provide a more accurate prediction for the presence of tumor cells than a simple CNN.

4.2 Discussion

As shown above, both CNN-RNN and transfer learning networks produce a strong prediction for the presence of tumor cells with testing accuracies of approximately 81%. Based on testing accuracies alone, medical professionals could use either model for prediction. In terms of processing time, however, we found that our transfer learning model took significantly longer to train than our CNN-RNN model. In training, the CNN-RNN only takes 14 minutes to train over 30 epochs, while the transfer learning model takes up to two hours to train and test on the same amount of data over 10 epochs. Therefore, if a quick prediction is needed from our model when implemented in the industry, we recommend using the CNN-RNN. On the other hand, the deep learning ability of ResNet could provide a more thorough learner for the given input if time permits.

4.3 Tumor Localization

We worked towards localizing the tumor cells in the given tissue images through using an explainable AI model known as LIME. We provided LIME with our model and a set of images that we wanted to interpret, and this framework provided us with the area of the image that was used for classification. Figure 10 below shows the results of four images classified by our CNN-RNN model.

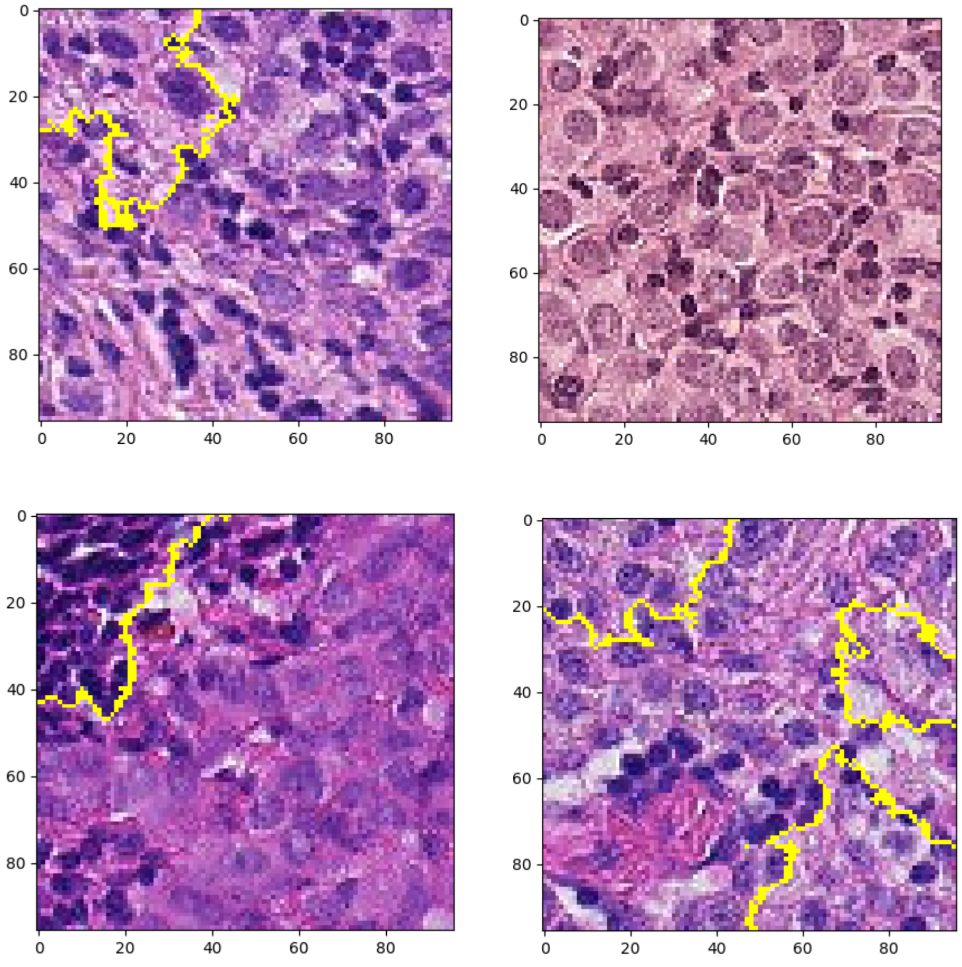

Figure 10: Outlined regions of our input images as given by LIME that our model used to classify.

The above images and the LIME framework are significant to solving our problem. The localization of tumors within tissue is a difficult task on its own, and using this framework and our model, medical professionals will also be able to learn how to identify the presence of tumor cells, along with predicting the presence of tumor cells.

5 Conclusion

In conclusion, we were able to achieve a maximum testing accuracy of 82% for the task of classifying whether tumor cells are present in a microscope image of lymph node cells using the CNN-RNN model. Our transfer learning ResNet50 model almost tied the CNN-RNN with an accuracy of 80% correct. Additionally, we were able to achieve our goal of applying XAI to this model. Through application of the LIME XAI framework to the CNN-RNN model, we were able to identify which parts of the image were important to the algorithm when making the classification. In the hands of a medical professional, information like this can be used to verify the classification of the algorithm.

REFERENCES

[1] Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In 2017 International Conference on Engineering and Technology (ICET) (pp. 1-6). Ieee.

[2] O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

[3] Li, Q., Cai, W., Wang, X., Zhou, Y., Feng, D. D., & Chen, M. (2014, December). Medical image classification with convolutional neural network. In 2014 13th international conference on control automation robotics & vision (ICARCV) (pp. 844-848). IEEE.

[4] Torrey, Lisa, and Jude Shavlik. "Transfer learning." Handbook of research on machine learning applications and trends: algorithms, methods, and techniques. IGI global, 2010. 242-264.

[5] Pan, Sinno Jialin, and Qiang Yang. "A survey on transfer learning." IEEE Transactions on knowledge and data engineering 22.10 (2009): 1345-1359.

[6] Quattoni, Ariadna, Michael Collins, and Trevor Darrell. "Transfer learning for image classification with sparse prototype representations." 2008 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2008.

[7] PyTorch. (n.d.). Retrieved April 23, 2021, from https://pytorch.org/hub/pytorch_vision_resnext/

[8] Moshkov, Nikita, et al. "Test-time augmentation for deep learning-based cell segmentation on microscopy images." Scientific reports 10.1 (2020): 1-7.

[9] Mou, L., Ghamisi, P., & Zhu, X. X. (2017). Deep recurrent neural networks for hyperspectral image classification. IEEE Transactions on Geoscience and Remote Sensing, 55(7), 3639-3655.

[10] Liang, G., Hong, H., Xie, W., & Zheng, L. (2018). Combining convolutional neural network with recursive neural network for blood cell image classification. IEEE Access, 6, 36188-36197.

[11] Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., & Xu, W. (2016). Cnn-rnn: A unified framework for multi-label image classification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2285-2294).

[12] Fleet, David, et al., eds. Computer Vision--ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I. Vol. 8689. Springer, 2014.