****

**Improving cancer image detection through AI/ML strategies to reduce translation invariant**

Jeff Bloom, Anna Cherkinsky, Andrew Petit, Joshua Malcarne

**Abstract**

Artificial intelligence models have achieved great success in cancer imaging. These models are often deep convolutional neural networks (DCNNs), and are most effectively used to supplement professional analyses. It is found that these networks have their drawbacks, as they are prone to overfitting and require massive amounts of labeled medical data to train. In this paper, we proposed a novel method of replacing the fully connected layers of DCNNs with pooling layers. This was performed using average pooling, global average pooling, max pooling, and max average pooling. Compared to more typical DCNN models, our proposed method can prevent overfitting tendencies without expensive hyperparameter tuning. Our model can also improve performance over more typical DCNNs while using smaller batch sizes. Experimental results on global average pooling demonstrate the effectiveness of the proposed method, outperforming the no-pooling model by a margin of nearly 5% for both recall and accuracy.

**Section 1. Introduction**  
Background  
 The application of artificial intelligence in the process of detecting cancer has greatly improved the diagnosis, monitoring, and treatment of many types of the disease [10]. However, despite providing improvement over more error-prone methods, such as manual detection, there remains significant issues related to the training and testing of machine learning models for cancer imaging [4]. In particular, the negative correlation between lack of clinical data for training and a resultant machine learning model’s inaccuracy is a massive dilemma. Misdiagnosed patients receive neither proper treatment nor a correct diagnosis until their cancer has become much more virulent, ultimately exposing them to higher cost of treatment, more rigorous treatment regimen, and increased risk of death. We can cite such a reference indicating AI can reduce false positives when compared to radiologists, who may easily miss subtle patterns. In a clinical study performed by Google, it was found that “In people with only one scan available, the AI outperformed all of the six radiologists who also examined the CT scans to assess risk of lung cancer. AI reduced the number of false positives by 11% and false negatives by 5%” [14]. Although we are not suggesting to replace pathologists, augmenting decision making by placing AI in the hands of pathologists will greatly reduce misdiagnosis. It is thus of extreme importance that machine learning models are innovated to grant increased training efficiency, cutting down on required training data whilst maintaining a high level of precision in clinical usage.  
 Medical datasets for cancer imaging are sparse, with even the largest datasets being vastly outsized by the massive online photo databases used to train image recognition AI [14]. This lack of ample training data for medical imaging AI is largely the result of how such data is validated. Clinical scans need to be manually analyzed and annotated by experts before they can be used to train CNNs [6]. A study presented at ISBI 2018 discusses one possible solution to this evident lack of data, exploring the potential for improved training efficiency of AI models given reduced datasets. The findings of this study observed that active learning techniques yielded a nearly 3% improvement in accuracy over random learning techniques for smaller training datasets in the context of cancerous tissue classification [6]. Similarly, our study aims to improve upon the inefficiency of training CNN’s whilst maintaining high precision rates by analyzing the performance of CNN’s augmented with Average Pooling, Max Pooling, Global Average Pooling, and Global Max Pooling techniques alike. Through these improvements, we hope that machine learning models for cancer image detection can be further enabled, improving cancer detection, lessening required analysis resolution and training data, and driving down medical costs by reducing manual radiological labeling and processing.

State-of-the-art applications/methods

In today’s medical industry, artificial intelligence is used rather extensively in regard to cancer identification. Typical approaches usually include some form of machine learning algorithm, with the main methods to classify being either artificial neural networks - specifically convolutional neural networks (CNNs) - or decision trees (DTs). Of these two approaches, CNNs are often regarded as the “gold standard” for this particular application, as decision trees usually perform worse and are more prone to changing wildly with variations in input data. However, CNNs are typically prone to overfitting - as they often show better accuracy while training than during testing. While overfitting can, to some degree, be addressed through optimizing hyper-parameters, this process tends to be extremely difficult and memory intensive. Grid search - an exhaustive search for discovering hyper-parameters - for instance, can take several days to complete depending upon the number of executors used for training and how large the dataset being trained on is. Bayesian optimization is more performant, but also scales poorly, becoming less and less efficient as the number of hyper-parameters for a neural network increases.

To address this issue, a new technique, Global Average Pooling (GAP), has started to become more widely adopted. This technique involves forgoing fully connected layers and instead replacing them with global average pooling layers. These layers force conformity between categories and feature maps which in turn makes interpreting results more meaningful. In addition, global average pooling layers require no additional hyper-parameters and act as a structural regularizer which in turn helps to prevent overfitting as no additional hyper-parameter tuning is required. Because of these added benefits, we anticipate that augmenting CNNs with this technique could prove particularly advantageous in predicting cancerous cells. As, in theory, the amount of time required to train and test should greatly decrease.

Novel Approaches

Most of the previous approaches to CNN used for cancer detection have been related to fully connected networks rather than GAP, Global Max Pooling, or other pooling techniques. Our novel goal is to develop and test AI learning algorithms using Average Pooling, Max Pooling, Global Average Pooling, and Global Max Pooling to enhance the efficiency of cancer diagnosis through digital imaging. We want to improve upon the techniques we found in our research. CNN techniques tend to use large amounts of memory and are not time-efficient due to the large number of parameters. The purpose of pooling techniques is to reduce the number of layers and amount of data used. [5]. This will help address some additional improvements to the accuracy of the model and potentially additional layers to remove based on our analysis. We plan to experiment with altering layer weights before using pooling to optimize the information processed. In order to control effects of variance and bias, we will use data augmentation of our data. If time permits, we will add a stretch goal to consider applying GAN (Generative Adversarial Networks) to improve on our ML model performance [3].

CNN has been used in research for AI learning algorithms to standardize cancer diagnosis through imaging. The approaches involved DCNN (deep convolutional neural networks) as well as RCNN (using regression) along with data manipulation methods in order to achieve more accurate and efficient results [1]. One study we came across used CNNs along with addressing cross-entropy loss, projection loss, label noise handling, and coupling classifier to diagnose multiple myeloma. Researchers encountered a problem with variable prediction accuracy results [2]. A study on using CNN for digital pathology used rotation and two by two strided average pooling to make predictions on histopathologic scans of lymph node sections [8]. The authors used rotations and global average pooling over p4 to maintain invariance in the rotation of CNN layers. However, their approach differs from ours because the authors first rotated the layers, and then used the GAP layer in order to control the variance. Images were split into 270 for training and 130 for testing. The authors use different hyperparameters on different networks. As can be seen by accuracy and area under the curve statistics, the approach depends on different hyperparameters, leading to > 87% accuracy. In conclusion, the modified CNN used by the PatchCamelyon research team has good performance. We will be conducting our experiment on the PatchCamelyon data. The PatchCamelyon benchmark is a new and challenging image classification dataset. It consists of 327.680 color images (96 x 96px) extracted from histopathologic scans of lymph node sections [8]. The paper which utilizes the PatchCamelyon dataset provides accuracy rates, areas under the curve, negative log likelihood, and different hyper parameters used [9]. We will be using the similar statistics for our experiment on PatchCamelyon data to compare the results. If we have time, we will be using the MedMNIST dataset to benchmark our results, as well. The MedMNIST v2 dataset consists of 12 pre-processed 2D datasets and 6 pre-processed 3D dataset [7]. The MedMNIST provides the prediction accuracy rates and areas under the curve (ROC) for the several provided datasets [7].

Using a curated dataset allows us to experiment on many model techniques without requiring additional and expensive processing power like GPUs, large VM’s to process, and reduces storage costs.

The techniques that follow will follow standard Machine learning practices which include:

1. Import the dataset
2. Normalize the data
3. Model Preparation via setup of InceptionV3 (TensorFlow).
4. Model setup includes, dropout, sequential, GAP tuning, etc.
5. Model create/compile/fit
6. Model Evaluate

The process from steps 4-6 will continue to iterate based on the Models predictive performance evaluation to detect cancer from the curated images.

Report Section outline:

In this report, we explain the related work which has been conducted in relation to our experiment, specifically pertaining to baseline sequential CNN, pooling methods, and GAN augmentation methods in Section 2. In Section 3, we explain our methodology regarding how we use the algorithms, display and explain the code we have implemented, and note the future methods we are considering approaching. In Section 4, we explain our experimental goal and display the results of our experiments so far. Section 5 is the discussion of future work.

**Section 2. Related Work**

Baseline sequential is used for the standard Convolutional Neural Network. The CNN consists of several nodes and tensors connecting the layers. The data and operations are passed in a computational graph, a technique called TensorFlow, which is what we are using to build CNNs for our experiment in Python. TensorFlow allows for freedom of dataset shapes, has high processing abilities, and provides a variety of learning algorithms and API, such as Deep Neural Networks and Recurrent Neural Networks. One of the gaps in the standard method is the lack of efficiency and a necessity to increase the number of iterations for data processing in order to achieve high accuracy results. A study on using CNN and TensorFlow on classifying images provides the following accuracy results for their CNN which has layers of input, convolutional layer, RELU, Pooling (for down-sampling), and FC (for class score). [11]

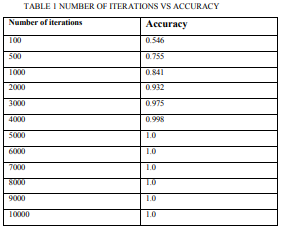


Figure: Accuracy for image classification using CNN with different number of iterations. [11]

The Global Max Pooling algorithm entails combination of all layers in a CNN, and reducing them to a layer with maximum values for each particular block of data.

The toy example below depicts the original dataset before and after it undergoes Max Pooling.

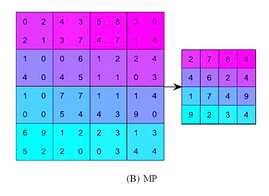


Figure: Example of Max Pooling; each batch is reduced using maximum values.

Each data block is reduced by taking the maximum value. [12]

Global Average Pooling reduces the convolutional layers by using the average of values. Global average pooling is explained, “For each feature map at the end of pooling layers, we take the average value of each feature vector directly maps to a category label or an output node” [12]. Since the original layers are replaced, the number of parameters needed for training is reduced. Due to the combination of the layers, the key features are still represented. Thus, global average pooling increases the efficiency of CNN. We will also be adding weights to the network to produce concrete metric results. Weights are placed in hidden layers of neural networks. Weights improve convergence rate. [12]

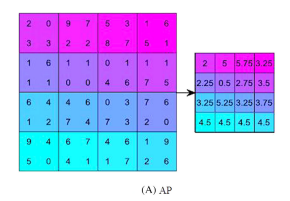


Figure: Example of GAP; each batch is reduced using average values. [12]

Augmentation involves transforming the image to add more images to the dataset. Since the same images are added (but are transformed) there is less variance among the data, meaning augmentation aids in reducing overfitting. Generative Adversarial Networks (GAN) are a form of augmentation which incorporates altered images which are not easy to distinguish from the original data. GANs typically have two layers, the generative and discriminative. The generative model creates “fake data '' which is altered from the original. The discriminator is trained to discriminate against the “fake data”. Hence, the resulting data is indistinguishable from the original. The newly generated data is then used in the classification process to reduce variance. [13]

**Section 3. Proposed Methodology**

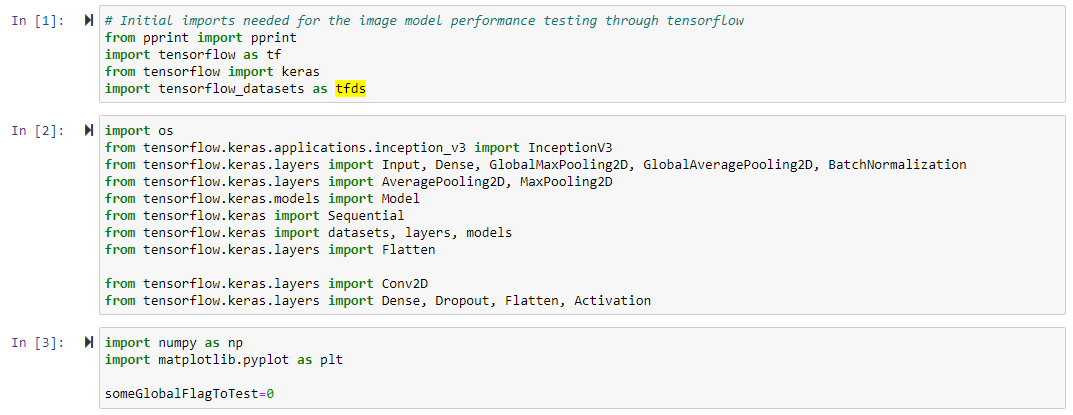
We will be implementing all programs and machine learning models using Python. Listed below are all models and programming frameworks that we will be using in this experiment:

* InceptionV3
* Keras (Tensorflow)
* Tensorflow dataset module
* Dense
* Model
* Average Pooling
* Global Average Pooling (GAP)
* Global Maximum Pooling
* Max Pooling

Our process begins by training the Inception V3 model, a highly regarded image recognition model. To do this, we start by pre-training the model and freezing all layers up to the bottleneck, when the layer has fewer neurons than the adjacent layers. Subsequently, we train the network and test it for accuracy and loss. Then, finally, unfreeze the data and compare results. Once evaluation completes with and without updated weights for the model, we retrain the model using data augmentation. Here we vertically flip the images and compare performance of accuracy and loss. Finally, we reevaluate the model using data augmentation techniques - such as flipping and image translation - then compare results.

Average pooling can provide a slight level of translation invariance; that is, by utilizing this technique, a machine learning model can recognize two images as the same regardless of pixels being shifted in different directions. In theory, this technique should improve model performance with respect to recognizing similar images. It works by blending less important pixels of an image together. Another technique, max pooling works slightly differently. This technique disregards less important pixels and retains only the max value. Global max pooling (GMP) and global average pooling (GAP) are very specific types of the aforementioned techniques, where the pool size equals the size of the input layer. Global max pooling will, for instance, take the max value over the entire image layer and calculate only the global max. It can, however, introduce and amplify noise into the model’s calculations. Conversely, global average pooling takes the global average instead of the global max. This results in a reduction in overfitting as there will be no additional parameters to optimize. In addition, GAP sums out spatial information and allows a resilient translation based on the input.

We use Keras, a software framework aimed at producing scalable machine learning models, backed by TensorFlow, an open source tensor manipulation framework, to implement our design. Additionally, we use numpy for array handling and matplotlib for plotting results. We begin by importing these libraries, as demonstrated in the image below:



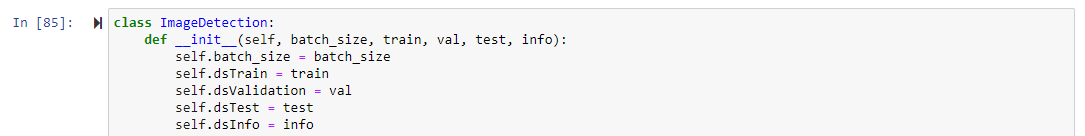
The above image shows a list of all libraries and imports that we are using.

We set up all of the pooling methods we will be using:

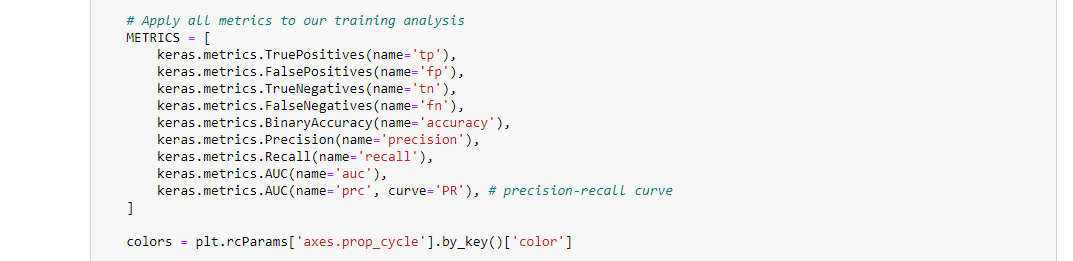


As listed above, we enumerated no pooling, global average pooling, global max pooling, average pooling, and max pooling as types.

We then implement an image detection class, which we use to initialize a batch size, training and test data and several other model specific attributes:



We defined the metrics we will calculate during processing of PatchChamelyon images. The metrics are listed below. We then also set up the color for our plots of these metrics.



We use the metrics described in the PatchCamelyon study to analyze model training and validation. Specific metrics we track are loss, precision, recall, and accuracy. We utilize matplotlib to plot these metrics to line graphs for easy interpretation. In addition, we display information for training recall, accuracy, precision, and loss to show performance improvements achieved by the machine learning model at the last epoch.

We define a function to plot accuracy, where we track the model accuracy, validation accuracy, loss, and validation loss and create graphs for these parameters. Loss is plotted for each epoch.



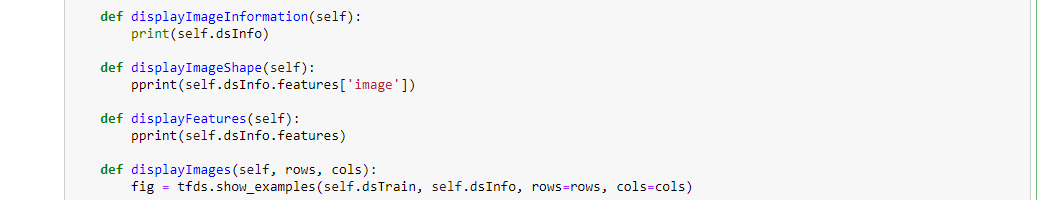
We made a function to plot all of the metrics we listed in the enumeration above, with epochs in the x axis.



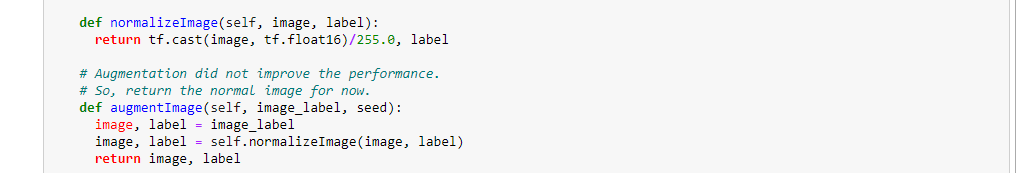
We make sure to round all metrics to the nearest hundredth to standardize the data comparison.



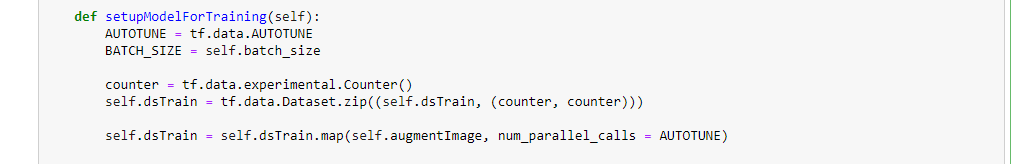
There are a few functions which display information about the image dataset



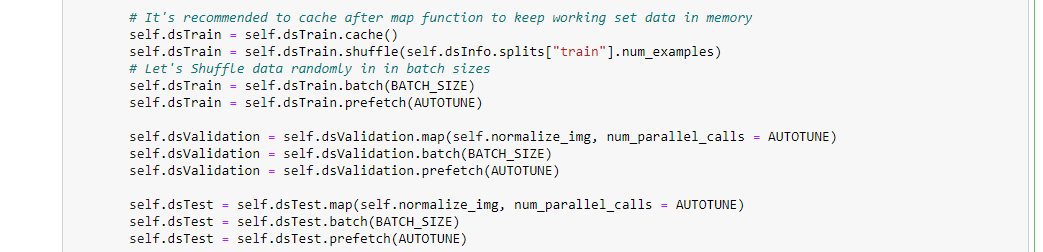
We normalize the image and use the results in augmentation. The results do not suggest augmentation improves performance. The augmentation types we performed were flip and rotate.



We autotune the data using tensorflow, and set up the data for training. The map function maps the image data with the label.



We cache the data to keep a working dataset inside memory. We then shuffle the data batches and prefetch it. We have the validation and test map, batch, and prefetch, which we perform for train, validation, and testing sets.

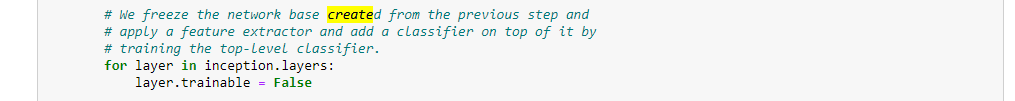


We then set up the model. We use the InceptionV3 for the model and to add imagenet weights to our layers. We then train the top-level classifier and extract features as well as add classifiers after freezing all other layers.





We set the trainable to false for each layer**.** The purpose of this is if the entire number of nodes is used, then the model learns from it, and blows up due to overfitting. This would take a very long time to process.

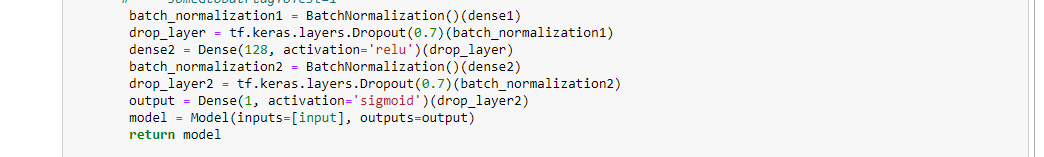
****

The Sequential() method adds a InceptionV3 layer followed by MaxPooling2D layers and ending with a Dropout layer to boost the models generalization capabilities.



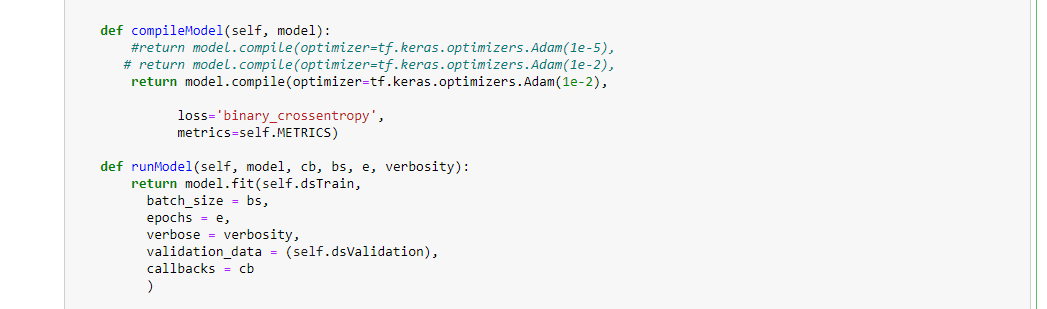
We then implemented cases for different pool types (GAP, Global Max Pooling, Average Pooling, and Max Pooling)



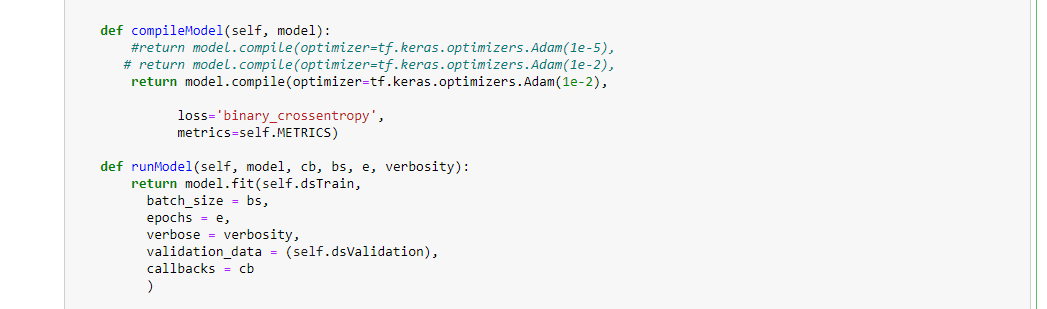


Above, we implemented the relu and sigmoid dense layers. To implement the process, we also add BatchNormalization into our layers to calculate the mean and variance of the layers input. This addition allows stabilizing the learning process and reduces the number of training epochs required to train deep networks. Adding more layers to the dense section can improve the models ability to classify the extracted features. The last dense layer activation returns between 0 and 1 and is similar to a 2 element Softmax activation.We use the Relu and sigmoid functions..

We made a function to compile the model, set the loss to binary cross entropy (cancerous or no), and the metrics we listed above.



We fit and run the model using the set up we provided above.



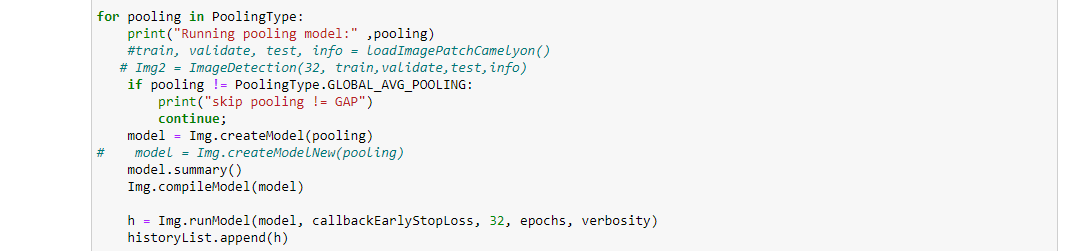
Then we load the data and call the program to obtain the results. The process took a little over two hours to finish running.



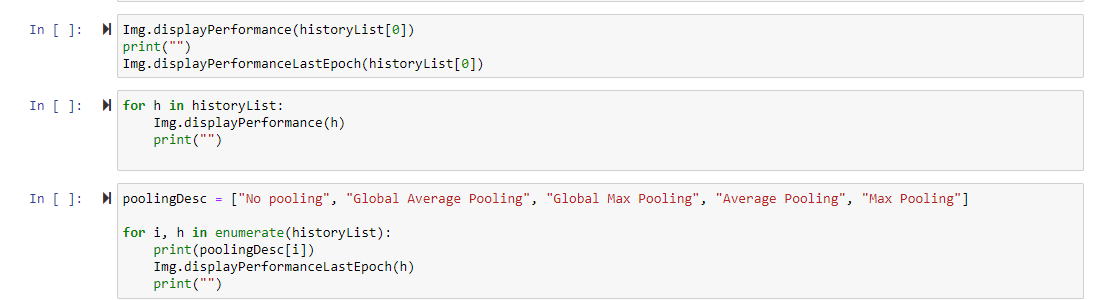
We used early stopping, which stops the processing early if there is no more learning after a certain number of epochs. Patience is the parameter which sets the number of epochs that would run without learning before the processing is stopped.



For each pooling type by compile and run the model

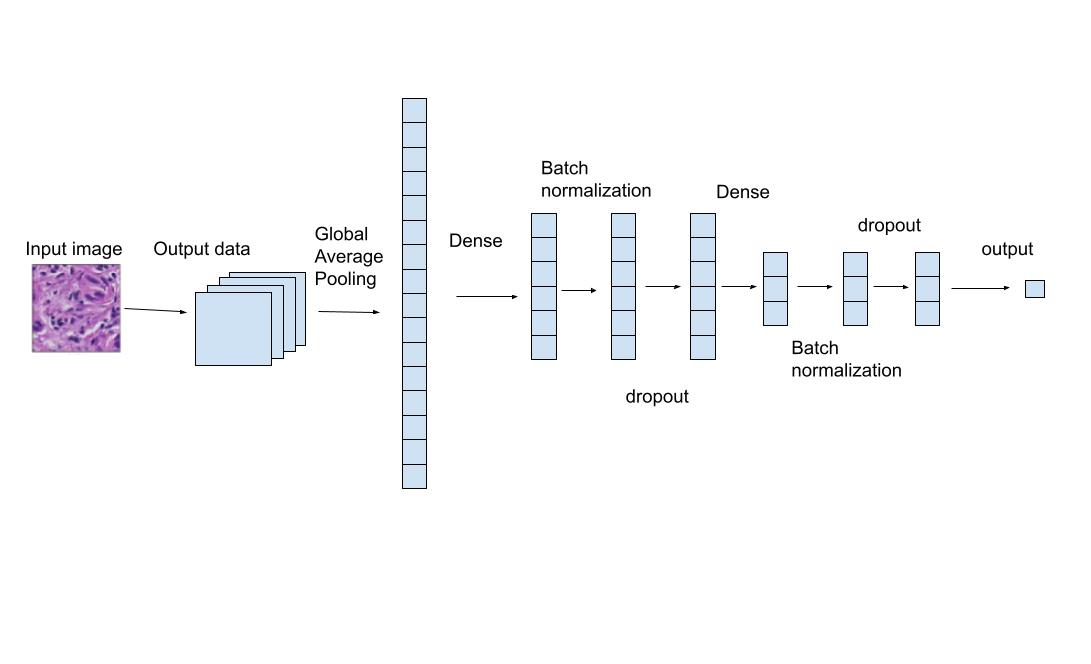


We display the performance we tracked in the historyList, and then produce the output.



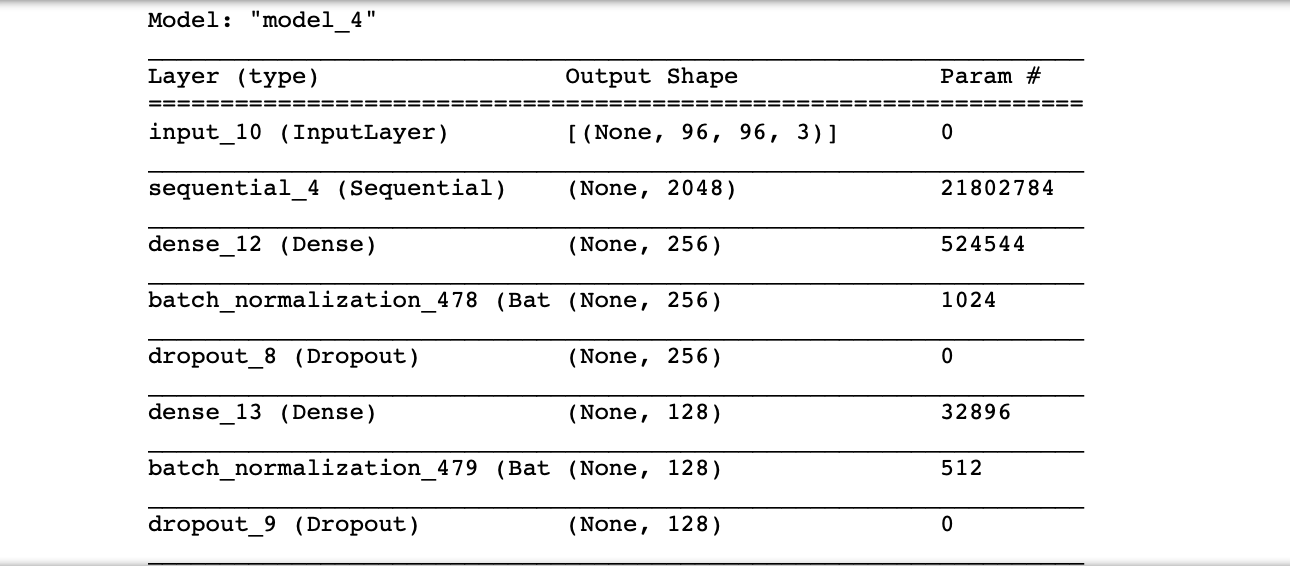
The final step was to display the performance epochs for each of the pooling types we used.

Below is a simple visualization of our architecture.



**Figure**. The architecture of our network with Global Average Pooling. After the GAP layer, the dimensions were compressed into average values for features which then got condensed.

The summary of our architecture is provided below:



We decided to maintain the original dataset without filtering out non-cancerous images, because our goal was to make the model learn from the original dataset. We also do not have enough medical knowledge regarding cancerous and none cancerous tissue, thus we could not cut down the images.

In the future, we hope to utilize GANs for further image augmentation. GANs, or generative adversarial networks, work by having two networks train against each other. One network, the so-called “generator,” will build fake images for the second network to train on. The second network will attempt to determine if images fed to it were authentic or faked by the “generator.” By utilizing GANs, we can create realistic images of cancer cells to aid with training our machine learning model. This could help solve a major issue in regards to lack of clinical data to train on, and further improve the accuracy of our machine learning model.

**Section 4. Experimental Results and Discussions**

First let’s describe the goals on what we want to achieve followed by how we will measure success in our performance of the model. The primary goal for an image classifier is to classify the image correctly. In order to identify and classify the image if we were human, we would identify images based on details and patterns.

For a model to classify an image in CNN, we will use what’s called downsampling.

This methodology is called pooling where you essentially shrink the image from the previous layer in a CNN and reduce its image pixel density.

We will consider four types of pooling and measure performance success against each approach.

Below are the four main approaches to pooling in CNNs:

* Average Pooling
* Global Max Pooling
* Global Average Pooling
* Max Pooling

Translation invariant is our goal, also, because the model will have the ability to recognize tumors regardless of where it is found in the image. One main method to achieve this level of model translation invariant is to apply pooling approaches.

Let’s begin with our dataset to use for our machine learning detection of cancer images. We will use PatchCamelyon for our analysis. The PatchCamelyon benchmark is a new and challenging image classification dataset. It consists of 327.680 color images (96 x 96px) extracted from histopathologic scans of lymph node sections.[9] As the datasets are well known, our success can be compared to various benchmarks such as those referenced in PatchCamelyon. The paper which utilizes the PatchCamelyon dataset provides accuracy rates, areas under the curve, negative log likelihood, and different hyper parameters used [8].

Next, let’s describe the setup required for image detection.We will pre-train the Inception V3 model and freeze all layers up to the bottleneck. Then train the network and test its accuracy and loss. Finally we unfreeze the layers and compare results. Once complete with evaluating with and without updated weights for the model, we retrain the model using data augmentation through our pooling approaches and compare performance of accuracy and loss.

A typical model takes on the following steps:

1. Load the data (tfdf.load())
2. Transform the data (split into test,train,validate)
3. Create the model (InceptionV3)
4. Build the model (model.compile())
5. Train and evaluate the model (model.fit())
6. Iterative process to Improve the model (comparing METRICS).

As described in the methodology section, we set up the Sequential() method with Inception V3 and MaxPooling2D. We also added BatchNormalization as a method to reduce variance at the input layers.

Here are the steps below:

* **Load the data**

(ds\_train, ds\_validation, ds\_test), ds\_info = tfds.load(

"patch\_camelyon",

split = ["train", "validation", "test"],

shuffle\_files = True,

as\_supervised = True,

with\_info = True,

# batch\_size=32,

data\_dir='/Users/jeffbloom13/WPI/CS534-AI/Project')

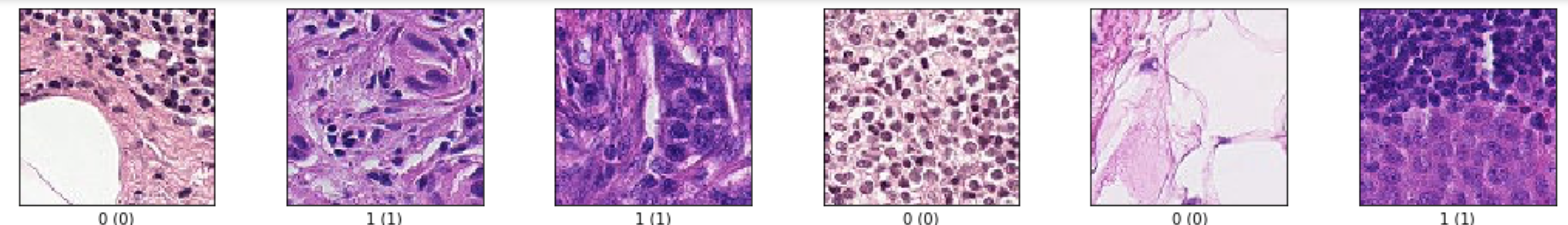
#data\_dir='/Users/jeffbloom13/WPI/CS534-AI/Project', try\_gcs=False)

* **Instantiate the ImageDetection Object**

Img = ImageDetection(64, ds\_train,ds\_validation,ds\_test,ds\_info)

* **Display a few images of the PatchChamelyon Dataset**

Img.displayImages(rows, columns)



* **Setup the Model for Training**

We build a model from patch\_camlyon 32k test, 26k training and 32k validation samples. As we see from the images above, binary classification labels describe presence of metastatic tissue(1= present, 0 =not-present). In addition, we set up the model to train our dataset with variable batch sizes, caching and mapping techniques to reduce runtime and compare performance results.

Img.setupModelForTraining()

Img.displayImageInformation()

tfds.core.DatasetInfo(

name='patch\_camelyon',

full\_name='patch\_camelyon/2.0.0',

description="""

The PatchCamelyon benchmark is a new and challenging image classification

dataset. It consists of 327.680 color images (96 x 96px) extracted from

histopathologic scans of lymph node sections. Each image is annoted with a

binary label indicating presence of metastatic tissue. PCam provides a new

benchmark for machine learning models: bigger than CIFAR10, smaller than

Imagenet, trainable on a single GPU.

""",

homepage='https://patchcamelyon.grand-challenge.org/',

data\_path='/Users/jeffbloom13/WPI/CS534-AI/Project/patch\_camelyon/2.0.0',

download\_size=7.48 GiB,

dataset\_size=7.06 GiB,

features=FeaturesDict({

'id': Text(shape=(), dtype=tf.string),

'image': Image(shape=(96, 96, 3), dtype=tf.uint8),

'label': ClassLabel(shape=(), dtype=tf.int64, num\_classes=2),

}),

supervised\_keys=('image', 'label'),

splits={

'test': <SplitInfo num\_examples=32768, num\_shards=8>,

'train': <SplitInfo num\_examples=262144, num\_shards=64>,

'validation': <SplitInfo num\_examples=32768, num\_shards=8>,

},

citation="""@misc{b\_s\_veeling\_j\_linmans\_j\_winkens\_t\_cohen\_2018\_2546921,

author = {B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, M. Welling},

title = {Rotation Equivariant CNNs for Digital Pathology},

month = sep,

year = 2018,

doi = {10.1007/978-3-030-00934-2\_24},

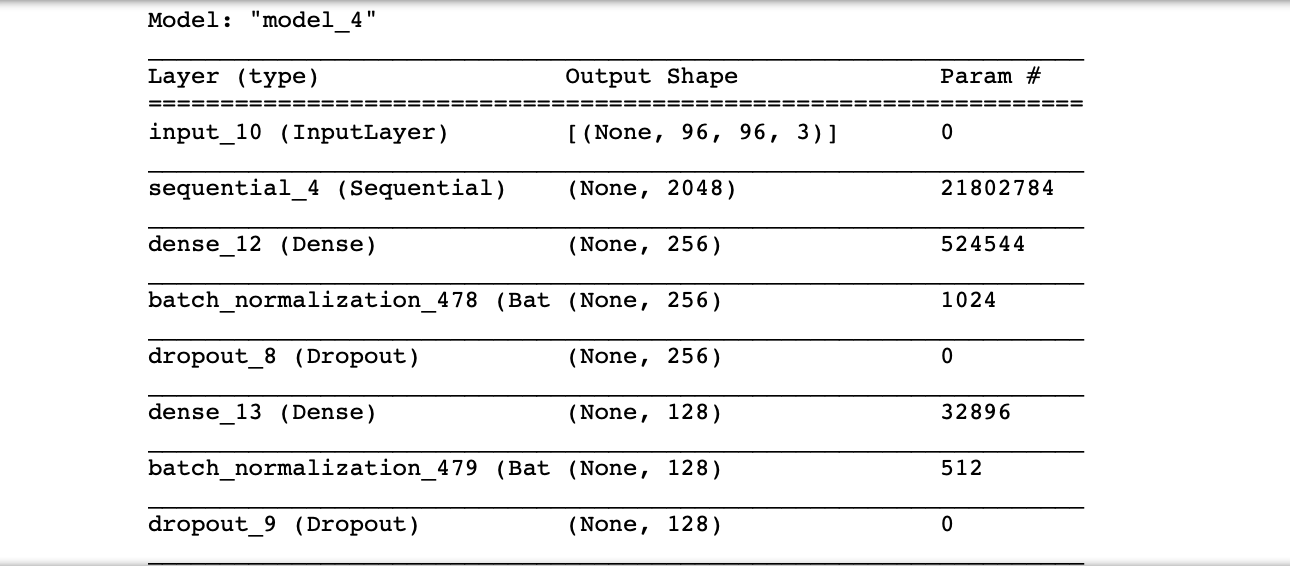
url = {https://doi.org/10.1007/978-3-030-00934-2\_24}

}""",

)

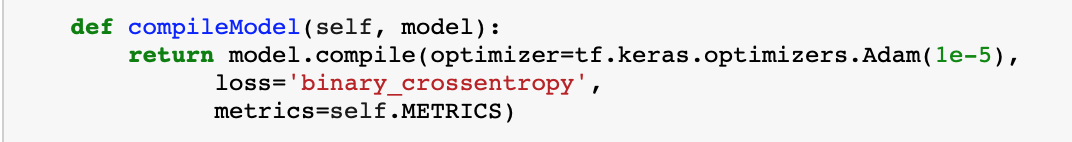
* **Create the Model**

model = Img.createModel()

model.summary()

* **Compile/Build the Model**

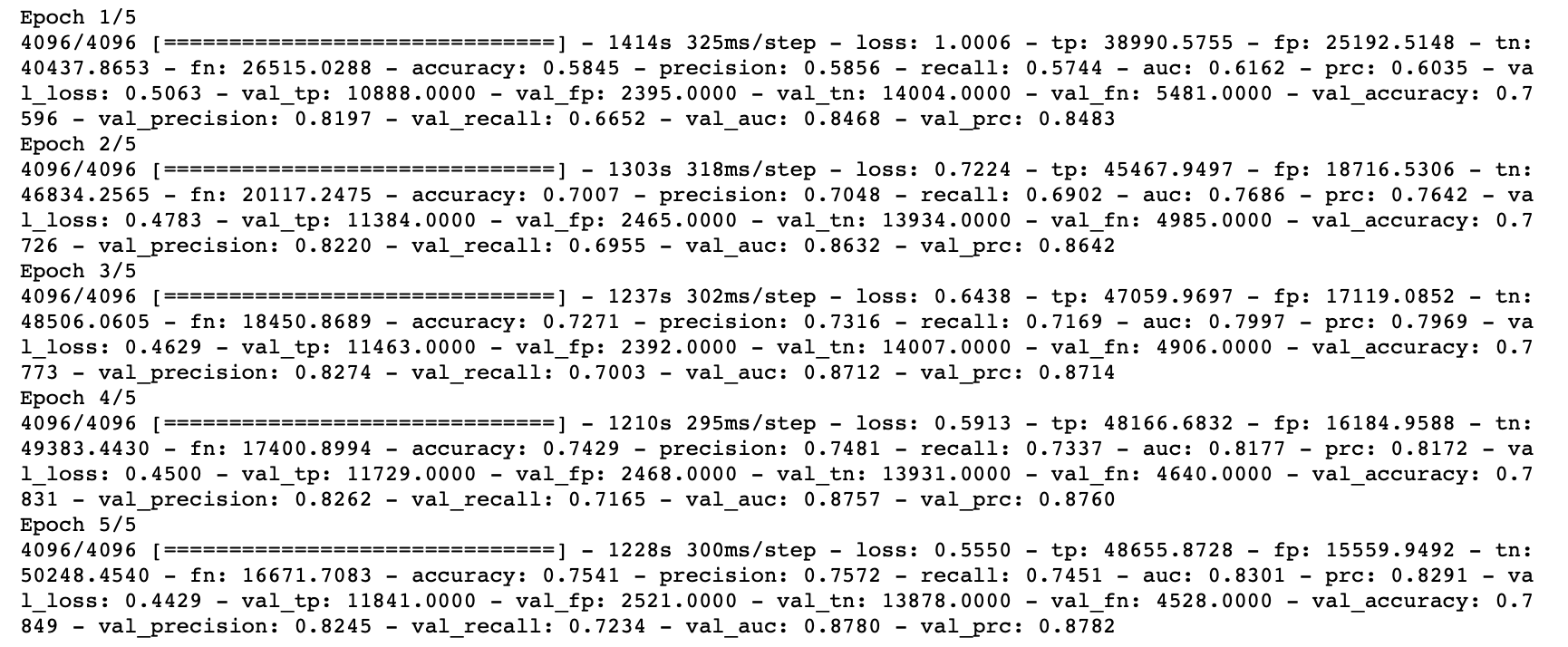
Img.compileModel(model)



* **Run the Model** (This executes a model.fit() )

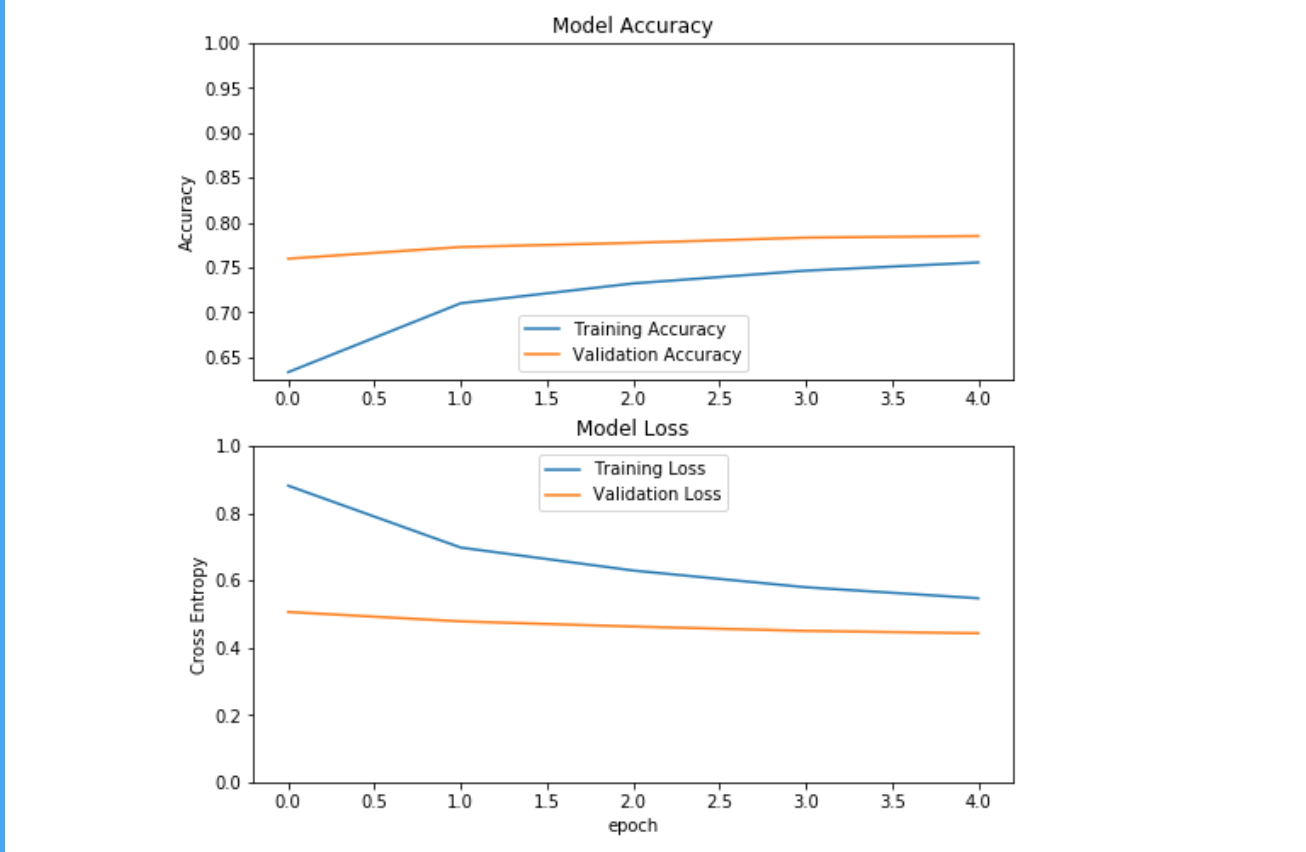
verbosity = 1

epochs = 5

history = Img.runModel(model, callbackEarlyStopLoss, 64, epochs, verbosity)

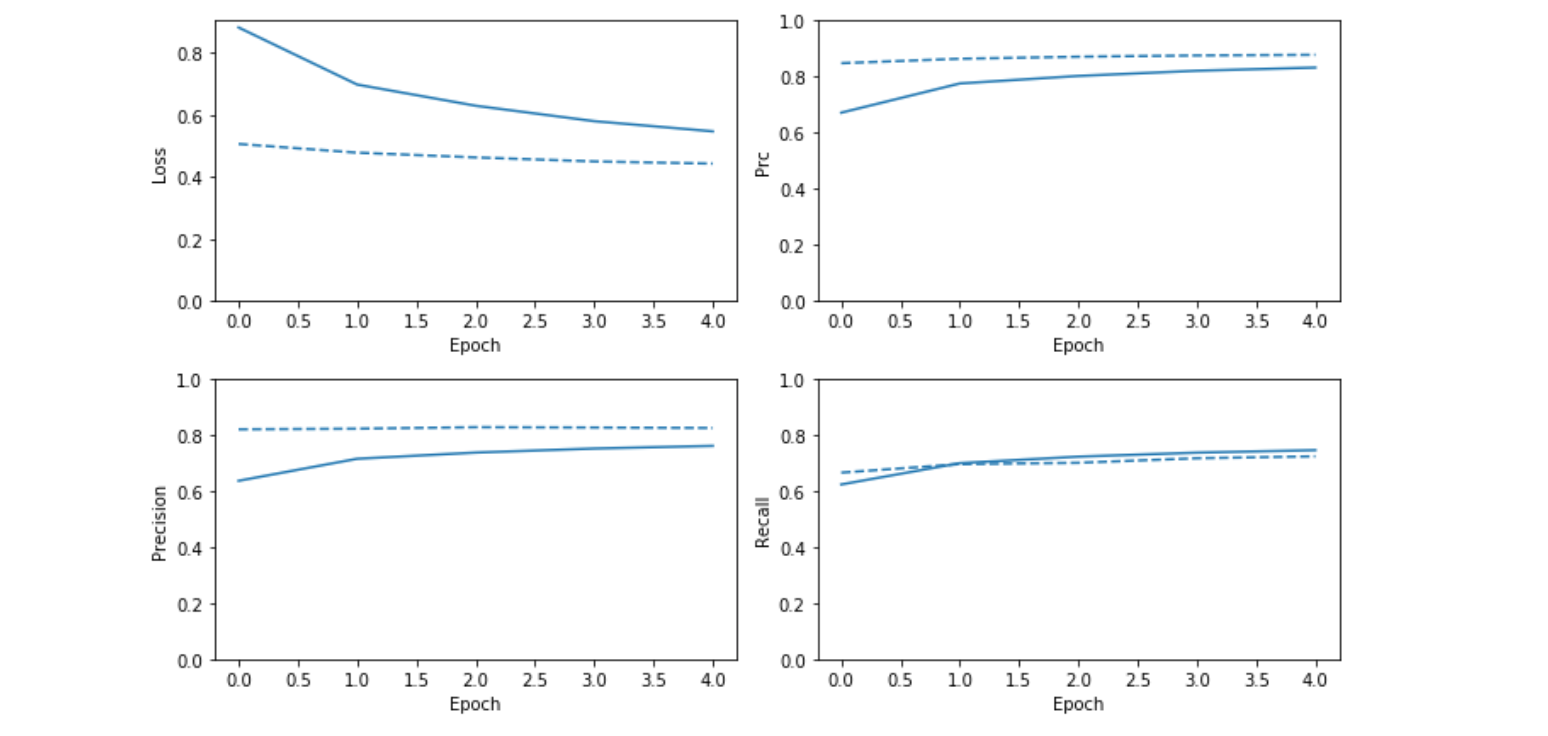
* **Plot the Model’s accuracy**

Img.plotAccuracy(history)



* **Plot Epocs vs precision, recall,**

Img.plotMetrics(history)



Based on the methodology above, we will iterate on batch size and all pooling methods to find the best performant model with and without weights. All comparisons will be based on validation precision recall and provide the corresponding plots.

The ideal outcome as seen in the above plots describe validation and training sets continue to converge over each epoch identifying the model does not overfit. We’ve also identified batch size plays an effect on the models performance whereas weights provide little to no improvement regardless of the models pooling layers. It’s likely a smaller batch size reduces the effects of change in it’s calculation versus larger changes may result in potential over fitting.

**Section 5. Conclusions and Future Work**

In summary, by utilizing various pooling techniques, we have seen notable increases in accuracy and recall without seeing explicit signs of overfitting - we have observed training and validation sets converge over each epoch, indicating that overfitting is minimal. The increases in accuracy and recall are especially evident in regards to the Global Average Pooling technique, where we noticed accuracy hit close to 80% and recall approach 90%. With this being noted, we have accomplished much of our initial goals of augmenting an existing approach for cancer detection to increase performance and reduce overfitting.

With the aforementioned being stated, however, there were several areas we think could provide additional benefit if explored in a future study. For instance, one aspect we identified as a potential means to improve model training would be to utilize GANs. Currently, labeled cancer data is somewhat difficult to find due to patient privacy concerns. With GANs, in theory we could produce additional images to train our network - essentially utilizing unsupervised learning to make our models fit better. In addition, our group unfortunately did not have the expertise to accomplish one of our stretch goals; to perform image segmentation as a means to further increase model performance. The datasets that we used to train our model ultimately did not indicate where in each image cancer was present, making it impossible for us to modify an image so that it always included relevant data. If we had the time, we might have followed up with oncologists to help us in the process, but a future study could attempt to do this and see if the results have any meaningful impact on model fitting.

**Works Cited:**

1. Agarwal, Yadav, A. S., Dinesh, V., Vatsav, K. S. S., Prakash, K. S. S., & Jaiswal, S. (2021). By artificial intelligence algorithms and machine learning models to diagnosis cancer. Materials Today : Proceedings. https://doi.org/10.1016/j.matpr.2021.07.088

2. Gehlot, Gupta, A., & Gupta, R. (2021). A CNN-based unified framework utilizing projection loss in unison with label noise handling for multiple Myeloma cancer diagnosis. Medical Image Analysis, 72, 102099–102099. https://doi.org/10.1016/j.media.2021.102099

3. Grnarova, Levy, K. Y., Lucchi, A., Perraudin, N., Goodfellow, I., Hofmann, T., & Krause, A. (2018). A domain agnostic measure for monitoring and evaluating GANs.

4. Hekler, Utikal, J. S., Enk, A. H., Solass, W., Schmitt, M., Klode, J., Schadendorf, D., Sondermann, W., Franklin, C., Bestvater, F., Flaig, M. J., Krahl, D., von Kalle, C., Fröhling, S., & Brinker, T. J. (2019). Deep learning outperformed 11 pathologists in the classification of histopathological melanoma images. European Journal of Cancer (1990), 118, 91–96. https://doi.org/10.1016/j.ejca.2019.06.012

5. Hekler, Utikal, J. S., Enk, A. H., Solass, W., Schmitt, M., Klode, J., Schadendorf, D., Sondermann, W., Franklin, C., Bestvater, F., Flaig, M. J., Krahl, D., von Kalle, C., Fröhling, S., & Brinker, T. J. (2019). Deep learning outperformed 11 pathologists in the classification of histopathological melanoma images. European Journal of Cancer (1990), 118, 91–96. https://doi.org/10.1016/j.ejca.2019.06.012

6. Folmsbee, Liu, X., Brandwein-Weber, M., & Doyle, S. (2018). Active deep learning: Improved training efficiency of convolutional neural networks for tissue classification in oral cavity cancer. 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), 770–773. https://doi.org/10.1109/ISBI.2018.8363686

7.Yang, Shi, R., Wei, D., Liu, Z., Zhao, L., Ke, B., Pfister, H., & Ni, B. (2021). MedMNIST v2: A Large-Scale Lightweight Benchmark for 2D and 3D Biomedical Image Classification.

8. Veeling, Linmans, J., Winkens, J., Cohen, T., & Welling, M. (2018). Rotation Equivariant CNNs for Digital Pathology. Medical Image Computing and Computer Assisted Intervention – MICCAI 2018, 210–218. https://doi.org/10.1007/978-3-030-00934-2\_24

9. Veeling, B. S., J. Linmans, J. Winkens, T. Cohen, M. Welling. (2018). *PatchCamelyon* (2.0.0) [Dataset]. TensorFlow. https://doi.org/10.1007/978-3-030-00934-2\_24.

10. Dlamini, Francies, F. Z., Hull, R., & Marima, R. (2020). Artificial intelligence (AI) and big data in cancer and precision oncology. Computational and Structural Biotechnology Journal, 18, 2300–2311. https://doi.org/10.1016/j.csbj.2020.08.019

11. Bandhu, & Roy, S. S. (2017). Classifying multi-category images using deep learning : A convolutional neural network model. 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 915–919. <https://doi.org/10.1109/RTEICT.2017.8256731>

12. Li, Wang, S., Fan, R., Cao, G., Zhang, Y., & Guo, T. (2019). Teeth category classification via seven‐layer deep convolutional neural network with max pooling and global average pooling. International Journal of Imaging Systems and Technology, 29(4), 577–583. <https://doi.org/10.1002/ima.22337>

13. Kollias, Cheng, S., Ververas, E., Kotsia, I., & Zafeiriou, S. (2020). Deep Neural Network Augmentation: Generating Faces for Affect Analysis. International Journal of Computer Vision, 128(5), 1455–1484. <https://doi.org/10.1007/s11263-020-01304-3>

14. Savage. (2020). How AI is improving cancer diagnostics. *Nature (London)*, *579*(7800), S14–S16. https://doi.org/10.1038/d41586-020-00847-2