Predicting Cytokine Responses From Images - Part 2!

Tempest Plott 2023-11-03

Problem Statement

Classification of microscopic images with machine learning has revolutionized biotechnology. However, the black-box nature of many machine learning approaches has left a gap between comfortable “knowns” of biologists and new “unknown” reasons for the classifications as generated by modeling. For example, some test drugs may cluster near control drugs and be listed as an exciting “hit” in a drug screen according to a clustering algorithm. But is the reason for the clustering really biological, or is it due to an artifact or a phenotype unrelated to the biology being studied?

Cytokines have been used as a known indicator of cellular activity for 50 years. By predicting cytokine production itself from images, rather than predicting black-box similarity to control drugs, I aim to bridge the gap between the old-school and the new-school. My goal is to add confidence in the power and usefulness of ML approaches in drug discovery, improve throughput of wet-lab approaches and analysis methods, and thereby reduce the cost of drug discovery.

The two questions addressed by the analysis presented in this report are -

1) Can the production of any of these 51 cytokines be predicted from 64x64 tiffs in CNNs or a fine-tuned ViTb32?

and

2) Does employing data augmentation improve the CNN performance?

For this analysis, I define predictable cytokines as those whose production can be predicted with at least 0.65 precision and recall. (In this analysis, chance would result in scores of 0.5.)

These questions require more context themselves, so please read the Part 1 project if you would like an explanation of the biology. (If you are currently wondering what on earth a cytokine is, I recommend reading Part 1.)

Data Collection

384 wells each containing 18,000 human white blood cells were plated. These were fixed and stained with Concanavalin-A, a fluorescent dye which paints the cell membrane. This shows the overall size and shape of the cells in the images.

51 cytokines were quantified for each of these wells. Each cytokine measure was expressed as log10-fold change relative to the mean of negative control wells and median-shifted to center at 0. This procedure is important for consistent communication between scientists and was therefore performed before handing off the data for further analysis.

Images of one field of each well were reduced to 64x64 pixel representations.

Cleaning the Data

Cleaning the features:

Pixel arrays were min/maxed to values of 0/1 to ensure that gradient descent steps would work appropriately across all images. (ie, it was ensured that brightness outliers would not be a detriment to learning.)

The dimensionality of the arrays was expanded to match tensorflow expectations. Later, the arrays were also expanded from one channel to three by repeating the information three times, to match the RGB format expectations of the ViT model. Proper data types were also enforced at this stage (numeric for features and categorical for labels.)

Cleaning the labels:

In the previous project, Each of 51 cytokine distributions was transformed into two bins with KBinsDiscretizer and labeled with 1 or 0 to indicate each well as either being a producer (1) or a non-producer (0) for each cytokine. Cytokines with seven or fewer producing wells were dropped from the experiment. Proper data types (numeric) were also enforced at this stage.

Figure 1: Histogram of experiment-wide normalized MMP-9 values.

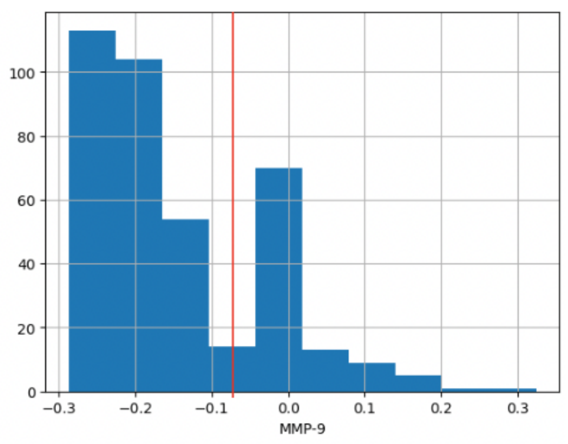


Figure 1: Example of KBinsDiscretizer splitting cytokine measurements into two bins. The Y axis is the number of wells and the X axis is the normalized cytokine value.

Exploratory Data Analysis

Features:

Augmentations were performed on the image arrays, and the arrays were shown in pseudocolor to confirm the augmentation appropriateness. Originally, rotations were employed, but these resulted in a vignette artifact. Thus, only flips were used in this final work to avoid visual artifacts. Horizontal, vertical, and transposition flips were used.

It is also noted that some tiffs are quite empty (such as the one shown below.) These empty images are not great candidates for this work, and a larger dataset should be employed in the future so that these empty images can more easily be dropped.

Figure 2: A successful horizontal flip, shown in pseudocolor.

Before: **After:**

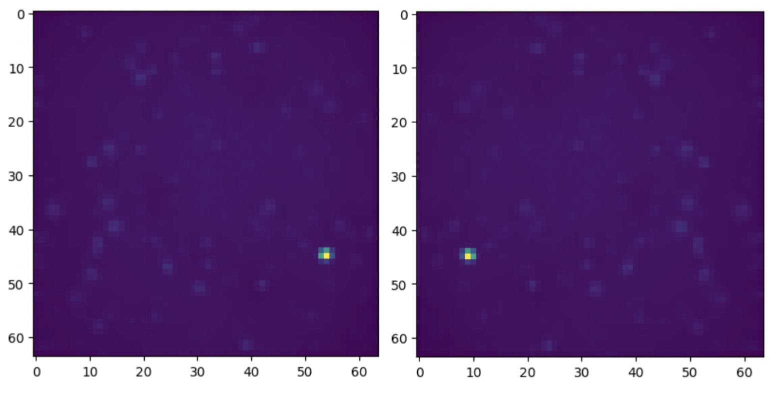


Figure 2: Well A01 has been appropriately flipped with no artifacts generated.

Labels:

The distributions of binarized cytokine production were observed. It was noted that there are many more non-producing wells than producing wells for essentially all cytokines, so a downsampling strategy was employed per-cytokine to remove non-producing wells at random to match the number of producing wells. This perfectly balanced the classes for the later modeling steps. Cytokines which had fewer than 15 wells after this downsampling process were dropped from the experiment. This left 27 cytokines to analyze.

It is important to note that some of the cytokines are independent of each other, while others are naturally produced in concert with each other. It was thus important to separate each cytokine into its own model to see if the model can truly learn to predict just that cytokine rather than "cheating" by using the signal from another cytokine as an input. Thus, it was also important to downsample each cytokine separately in the data preparation stage.

Figure 3: Histogram of experiment-wide binarized cytokine labels.

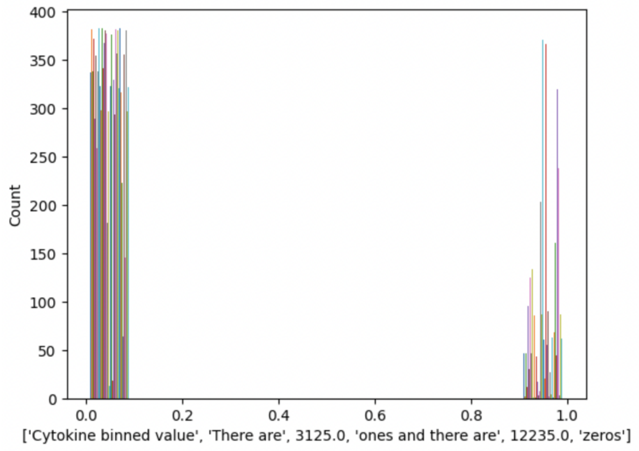


Figure 3: There was about a 4:1 ratio of wells labeled 0 to wells labeled 1 before downsampling 0s for each cytokine at random.

Data Wrangling

For the downsampled cytokines, labels were fetched and expanded to match the size of the augmented feature set as appropriate, and feature and label arrays were formatted to match tensor flow expectations.

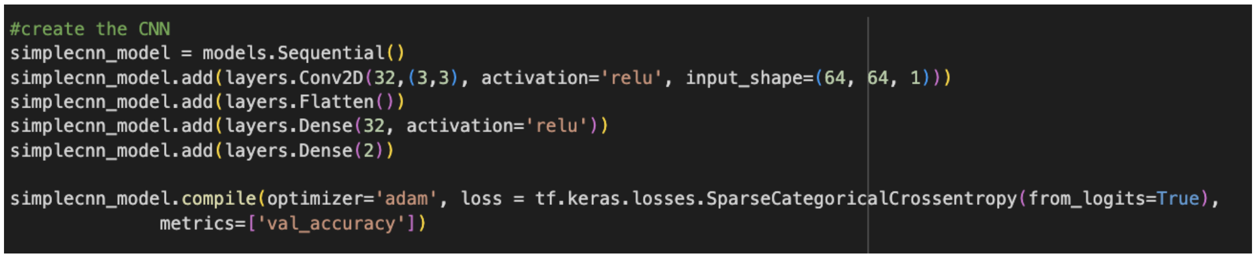
Each cytokine has its own model, run two-fold. The history and performance of each model was stored in a list of lists for display and summarization.

Modeling

A summary of the five models used is in the table below.

Model	**Number of Trainable Parameters**
Wider CNN	3,936,674
Deeper CNN	519,010
Simple ViT	769
Fine-Tuned ViT	240,129
ViT 500k	568,321

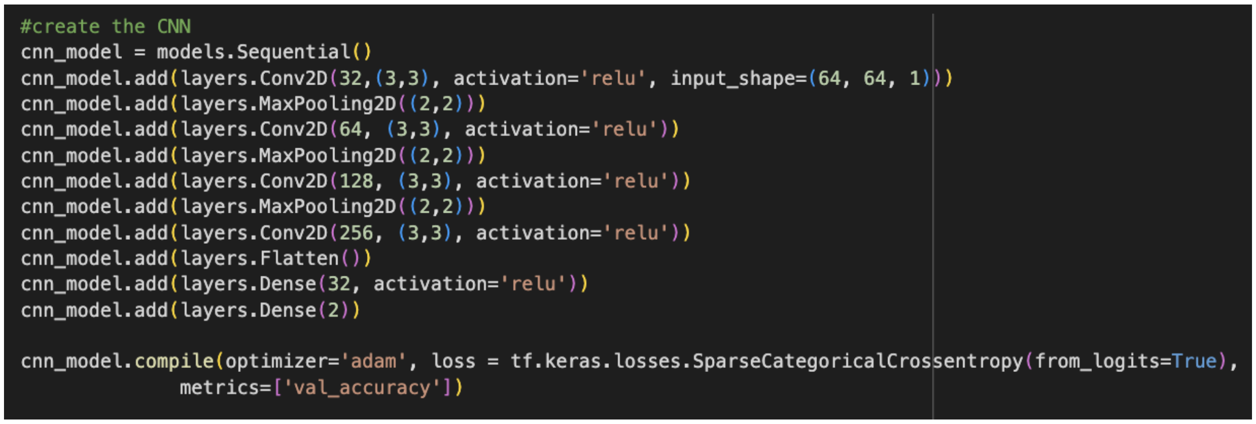
The wider CNN model trained independently on each cytokine separately is as follows:



```
#create the CNN
simplecnn_model = models.Sequential()
simplecnn_model.add(layers.Conv2D(32,(3,3), activation='relu', input_shape=(64, 64, 1)))
simplecnn_model.add(layers.Flatten())
simplecnn_model.add(layers.Dense(32, activation='relu'))
simplecnn_model.add(layers.Dense(2))

simplecnn_model.compile(optimizer='adam', loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
            metrics=['val_accuracy'])
```

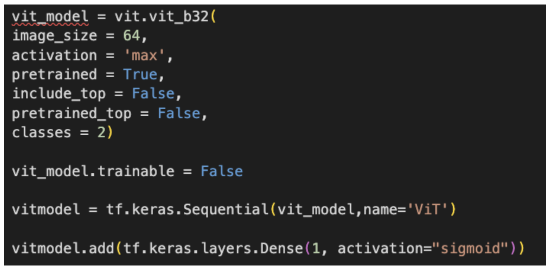
While the data are not particularly sparse (half of the labels are 0 and half are 1), the “Sparse Categorical Cross-entropy” loss function performed best and is not overtly inappropriate for use with this data.

The deeper CNN model is as follows:

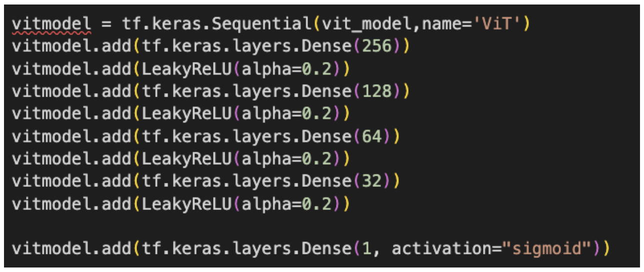


```
#create the CNN
cnn_model = models.Sequential()
cnn_model.add(layers.Conv2D(32,(3,3), activation='relu', input_shape=(64, 64, 1)))
cnn_model.add(layers.MaxPooling2D((2,2)))
cnn_model.add(layers.Conv2D(64, (3,3), activation='relu'))
cnn_model.add(layers.MaxPooling2D((2,2)))
cnn_model.add(layers.Conv2D(128, (3,3), activation='relu'))
cnn_model.add(layers.MaxPooling2D((2,2)))
cnn_model.add(layers.Conv2D(256, (3,3), activation='relu'))
cnn_model.add(layers.Flatten())
cnn_model.add(layers.Dense(32, activation='relu'))
cnn_model.add(layers.Dense(2))

cnn_model.compile(optimizer='adam', loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
            metrics=['val_accuracy'])
```

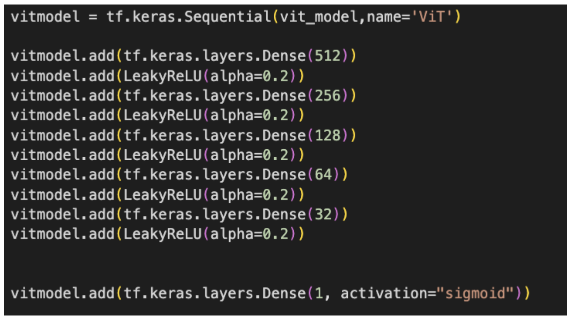
The simple ViT model (which has a top of merely one activation layer) a more fine-tuned version of the ViT with similar top layers as the deeper CNN model , and a version of a fine-tuned ViT with a similar number of trainable parameters as the deeper CNN are as follows:



```
vit_model = vit.vit_b32(
image_size = 64,
activation = 'max',
pretrained = True,
include_top = False,
pretrained_top = False,
classes = 2)

vit_model.trainable = False

vitmodel = tf.keras.Sequential(vit_model,name='ViT')

vitmodel.add(tf.keras.layers.Dense(1, activation="sigmoid"))
```



```
vitmodel = tf.keras.Sequential(vit_model,name='ViT')
vitmodel.add(tf.keras.layers.Dense(256))
vitmodel.add(LeakyReLU(alpha=0.2))
vitmodel.add(tf.keras.layers.Dense(128))
vitmodel.add(LeakyReLU(alpha=0.2))
vitmodel.add(tf.keras.layers.Dense(64))
vitmodel.add(LeakyReLU(alpha=0.2))
vitmodel.add(tf.keras.layers.Dense(32))
vitmodel.add(LeakyReLU(alpha=0.2))

vitmodel.add(tf.keras.layers.Dense(1, activation="sigmoid"))
```



```
vitmodel = tf.keras.Sequential(vit_model,name='ViT')

vitmodel.add(tf.keras.layers.Dense(512))
vitmodel.add(LeakyReLU(alpha=0.2))
vitmodel.add(tf.keras.layers.Dense(256))
vitmodel.add(LeakyReLU(alpha=0.2))
vitmodel.add(tf.keras.layers.Dense(128))
vitmodel.add(LeakyReLU(alpha=0.2))
vitmodel.add(tf.keras.layers.Dense(64))
vitmodel.add(LeakyReLU(alpha=0.2))
vitmodel.add(tf.keras.layers.Dense(32))
vitmodel.add(LeakyReLU(alpha=0.2))


vitmodel.add(tf.keras.layers.Dense(1, activation="sigmoid"))
```

The vit_model trainability is set to false so that only the fine-tuning top layers are trained, rather than the entire generalized object detection base of the model. “Binary cross-entropy” is recommended as the default loss function for binary data, and in this case it did perform better than some other options which were used, such as mean square error. Leaky ReLu activation is used to maintain as much information as possible from the data at each step and finally sigmoid activation is used to decide if the label prediction should be “1” or not.

During the fit stage of modeling, excessive over-fitting was mitigated by the use of early stopping. Training was set to stop if the loss metric did not improve for three epochs in a row. However, that condition did not ever materialize.

Results

CNN Performance

The CNN models performed better than the approach in Part 1 of this work, which used embeddings from high-resolution images and an XGBoost model to successfully predict cytokines with an average F1 score of about 0.75. The CNN models produced 26 predictable cytokines with an average F1 score of about 0.8. The wide CNN model missed only one cytokine which was predicted by the embeddings - IL-16.

Figure 5: All Predictable Cytokines (from either CNN model) and their F1 scores

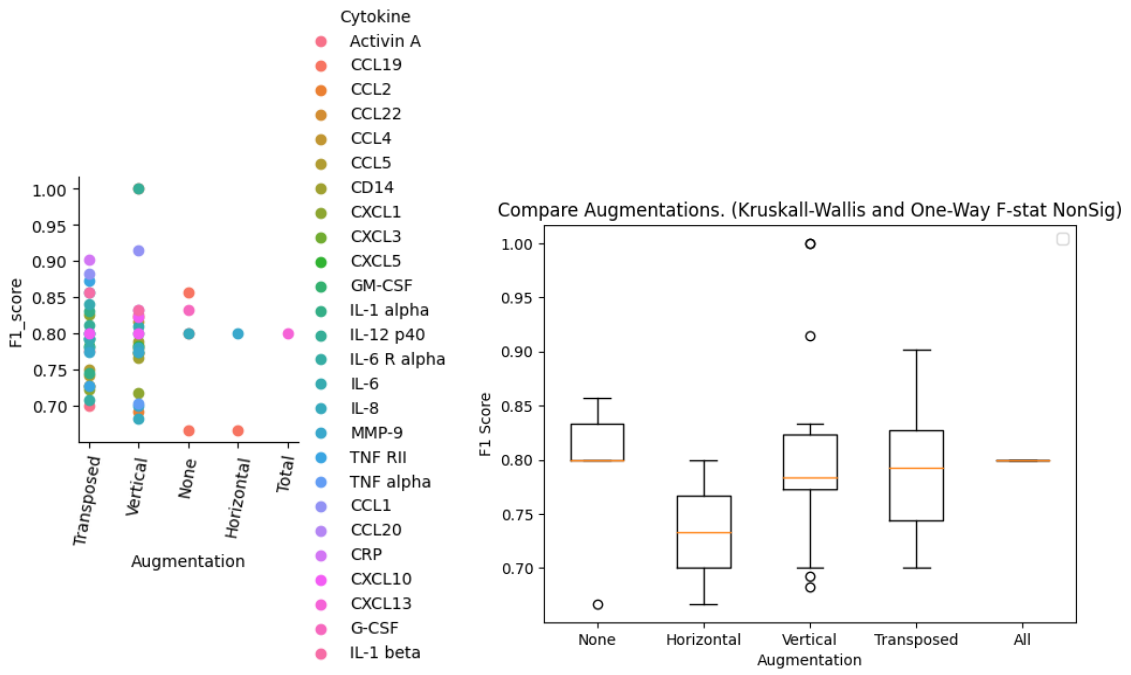


Figure 5: A scatterplot and a boxplot of the F1 scores for all predictable cytokines are shown, with scores grouped by the augmentation strategy for each tested dataset.

While it may appear to the eye that the vertical augmentation performed the best, no statistical test supported the hypothesis that the distributions are truly different. Since there are five augmentation groups and the length of the series of F1 scores for each group should be interpreted as an important factor (ie, differences in sample size are important and should not be assumed to be equal for the test), non-parametric Welch's ANOVA was used. However, regular ANOVA and the Kruskal-Wallis test were also employed for thoroughness. None of these tests showed a statistically significant p-value anywhere near 0.05. Perhaps with a larger dataset, a stat sig trend would make itself known. In conclusion, the augmentations used here did not benefit the CNN model, but neither did they harm anything.

However, the wider CNN model did perform significantly better than the deeper CNN model. The wider CNN model produced both more predictable cytokines and a higher average F1 score. The result of a non-parametric Welch's T-Test for these two models was a p-value of 4e-08. The distribution of predictable cytokines and their F1 scores is shown below.

Figure 6: Distributions of performance for each CNN model

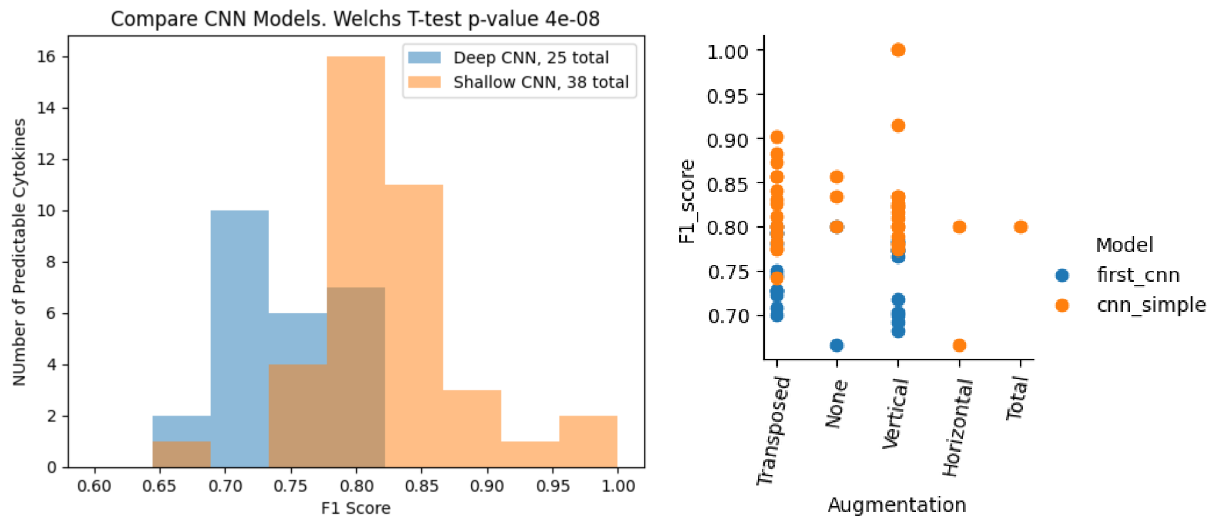


Figure 6: In the above figure, "first_cnn" refers to the deep CNN and "cnn_simple" refers to the wide CNN.

ViT Performance

None of the ViT models successfully predicted any cytokines in any fold for any data augmentation strategy. In general, the ViT models tended to guess mostly positive or negative to rely on chance to maximize the metric rather than learn anything. The ViT model was also applied to the unaugmented data set and raw pixel arrays (not normalized to a range of 0 and 1) for thoroughness, but neither of these improved the performance .The confusion matrix for an example cytokine which was predictable by the CNN is shown below.

Figure 7: Example of bad ViT performance - GM-CSF, which was predictable with the CNN approach

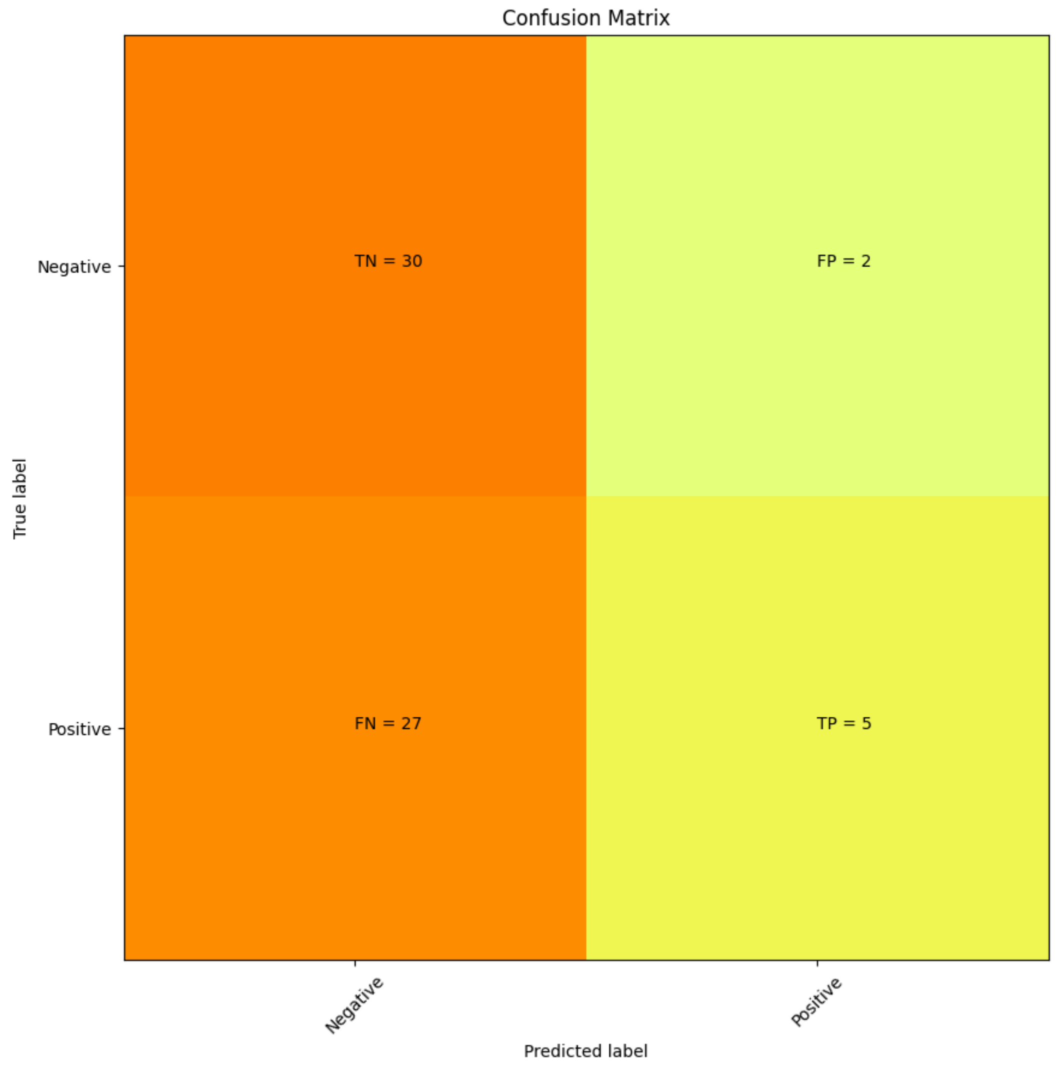


Figure 7: GM-CSF precision = 0.71, recall = 0.15

Discussion

The reason for the poor ViT performance is likely due to the quality of the input images. 64X64 pixel tifs are very low resolution compared to the massive number of images which pre-trained ViTb32. (They are also lower resolution than the images which resulted in the embeddings of Part 1 of this project.) Also, ViTb32 was trained on RGB images. The strategy employed here of simply repeating one channel three times to masquerade as an RGB image was unsuccessful. The CNN, however, had no prior notions of what images and features therein should look like, and thus performed much better.

For the case of the CNNs, it is good to know that more augmentation does not necessarily improve performance. Augmenting datasets is probably not an efficient first step to take - it is better to explore the performance on the true data first, look for weak points, and determine if augmentation is needed.

I would also like to mention some explanation for the curious as to why the wide CNN, which has only one layer, has so many more trainable parameters than the deeper CNN. Since a fully connected layer follows the convolutional layers, removing the convolutional layers results in parameterization of the entire image. This can be understood via the mathematical formula for parameter calculation. But first - in plain english, the convolutional layers are simplified abstractions of the image. Adding more of them before the final layer actually reduces the trainable parameters in the case of my model architecture.

An example of how this occurs follows:
To calculate the parameters between a fully connected layer and a convolutional layer, use this formula:

((Conv layer height * width * channel) + 1) * units in FC layer

Consider reducing this two-convolutional layer model to only one.

```
64 * 64 * 3    ---->    32 * 32 * 8    ---->    64 * 1
64 * 64 * 3    ---->    64 * 1
```

Initial parameters: *(32 * 32 * 8 * 64) + (8 * (1 + (2 * 2 * 3)))= 524288 + 104* = 524,392
Parameters after removing a convolutional layer: *64 * 64 * 3 * 64 = 786,432*

The number of trainable parameters increases by more than 50% by reducing the depth by just one layer.

Future Research

There are a few things I could do to improve the performance of these models.
I should repeat this work with a larger dataset than n=384.
Each CNN for each cytokine should be individually tuned for best performance. This includes many potential changes.
I could also use a different pre-trained model, such as EfficientNet, rather than ViTb32.
It is likely that a deeper transfer process and a larger dataset would be needed to successfully transfer any pretrained model to such niche data.

This work can be used as 1) a proof of concept which validates the ability of the current imaging laboratory approach to capture real biological signal, 2) a note that low quality data can be useful, and 3) a test case showing newer, fancier methods do not guarantee improvement of results. You can't expect a ViT trained on photographs that contain no images of cells to easily transfer to such a specific type of image.

Thanks

Gratitude to Noor Hussain and Ahmed Hosny for their guidance in performing this analysis.

Code from this post was used/adapted to create more visually appealing confusion matrices.

Thanks to these posters for explaining more about the CNN architecture.