



Predicting Cytokine Responses Via CNN and ViT models



Tempest Plott
2023



The Problem:

Obtaining cytokine readings is an important part of biological drug screening, but is time-consuming, expensive, and risky.

The Solution:

If it is possible to predict cytokine response from images, then drug discovery can happen faster, with less risk, and with greater ease.

The Mission:

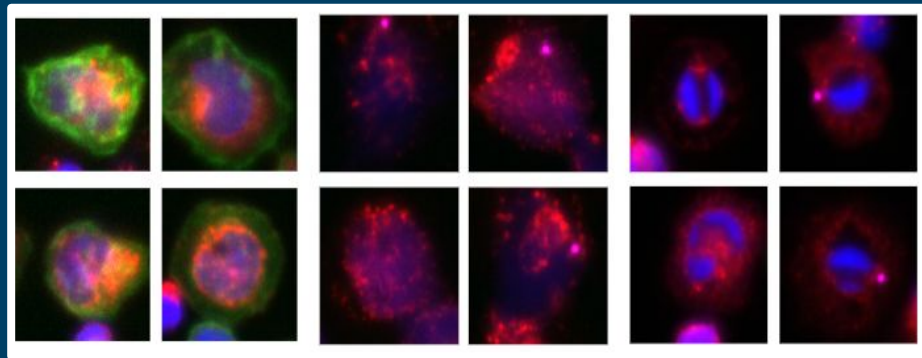
ML has revolutionized biotech. But are the black-box similarities of “hits” and control drugs really biological, or are they due to a phenotype unrelated to the biology being studied?

By predicting cytokine production itself, 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 reduce the cost of drug discovery.

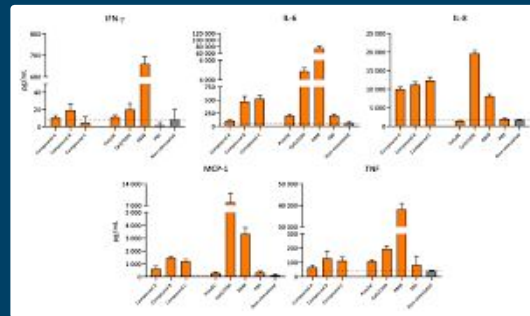
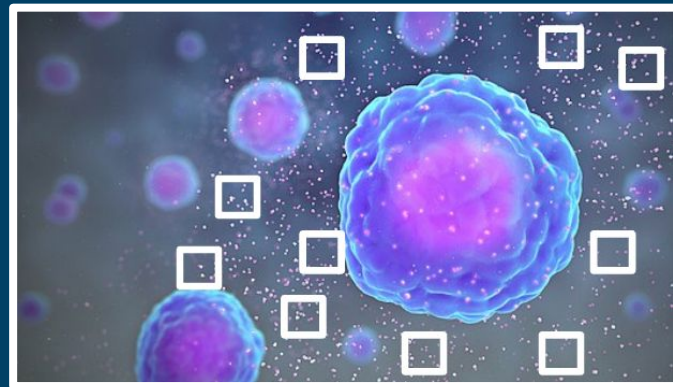
Background Information:

Real High Content Microscopy Images

Cytokine Schematic and Data Example



Actin, mito, nucleus



The Constraints:

Use previously collected low-resolution image thumbnails from a previous experiment.

The Problem, in Context:

Can CNNs or ViTs predict cytokine production better than the previous Part 1 approach - using image embeddings of high-resolution images?

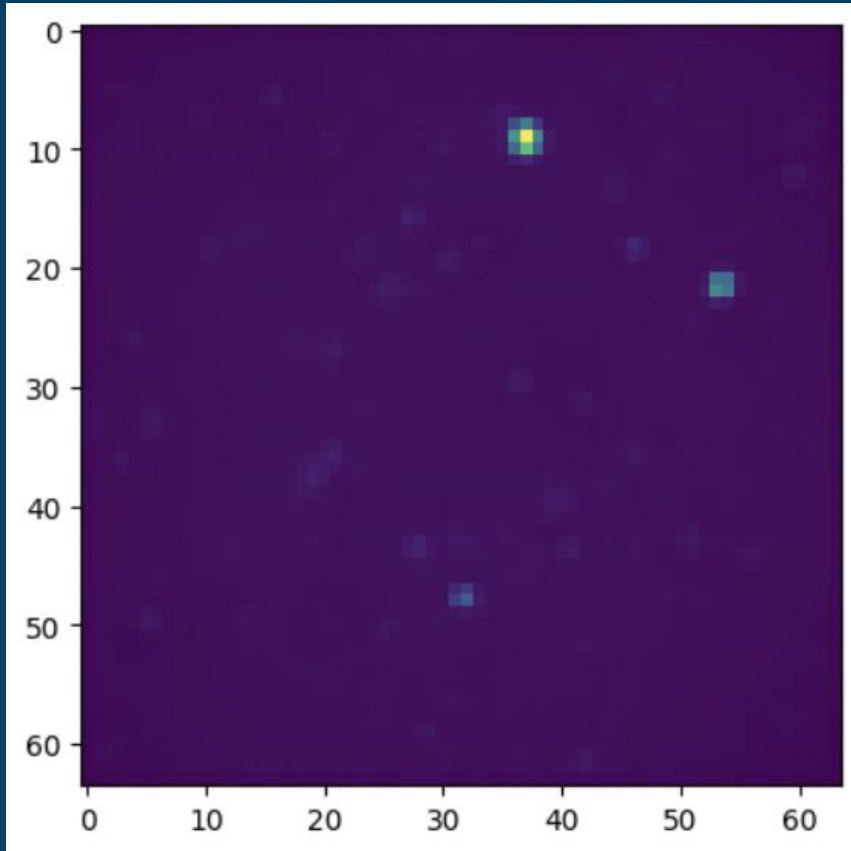
Are low-resolution images usable?

Does augmentation improve performance of these models?

The Data:

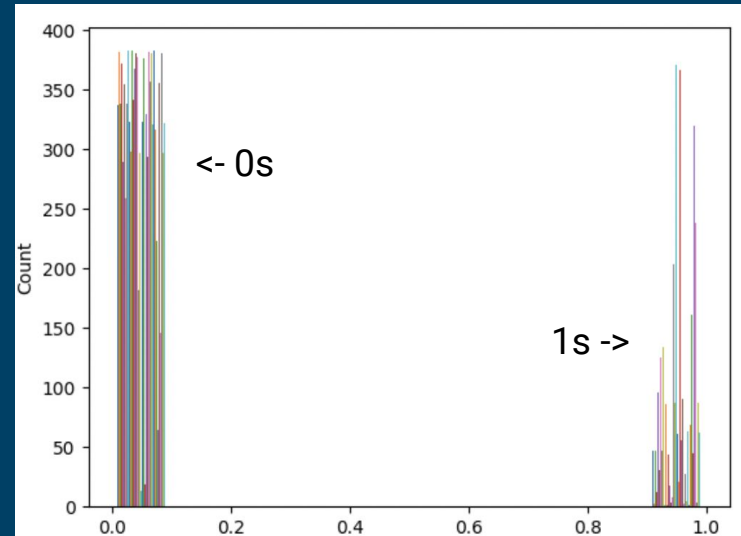
Use a proof-of-concept subset with Con-A
Embeddings, 64X64 pixel tiffs, 50 cytokines, and
only one plate.

Well Images (Features)



Binarization and balancing of Well-Based Cytokine Readouts (Target)

		Name	APRIL	Activin A	Amphiregulin	CCL1
plate	well					
	A01		0.0	0.000000	0.0	0.000000
	A02		0.0	-0.067838	0.0	-0.023606
	A03		0.0	-0.067838	0.0	-0.425776



Data Wrangling


Remove data that is not useful and ensure appropriate data types are enforced for later analysis.

- ❖ After binarizing and class-balancing cytokines, those with fewer than 15 remaining wells were dropped. (This is 7 positive response wells.)
- ❖ Applied 4 data augmentation strategies (flips) to arrays and enforce appropriate array shapes and data types for modeling architecture.
- ❖ Attach downsampled embeddings and labels into independent dataframes for each cytokine. Loop through these for two-fold modelling.

Models tested

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

Models tested



```
#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'])
```

Every model was tested two fold on each augmented dataset.
Early stopping was employed.

Models tested

```
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(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"))
```

Results

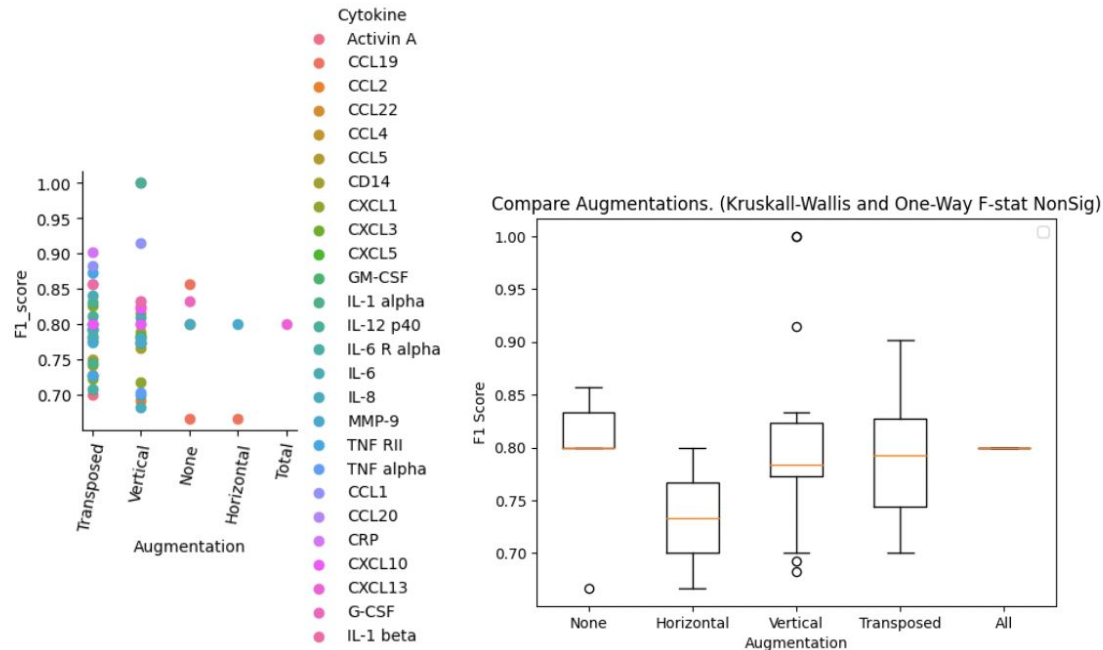


Figure 5: A scatterplot and a boxplot of the F1 scores for all predictable cytokines are shown, with scores grouped by the augmentation strategy for each tested dataset.

Success! The CNN models predict many of the cytokines. There is no stat sig difference in the performance of the 5 different augmented datasets.

CNN outshines Part 1 with twice as many cytokines and higher F1 score.

Results

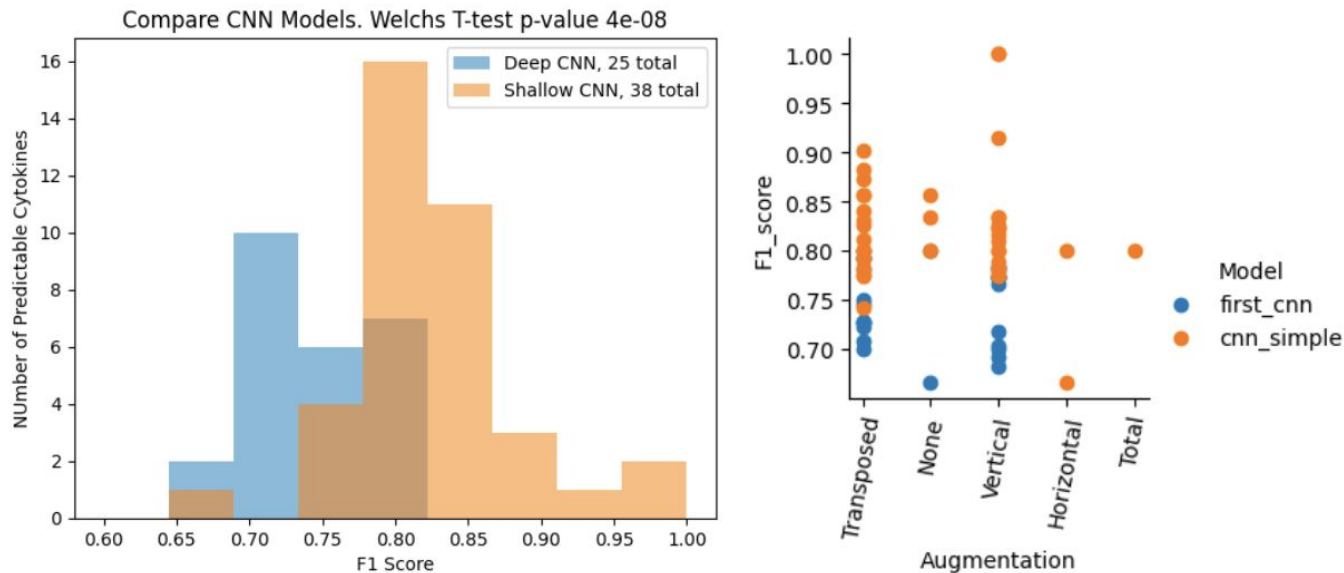


Figure 6: In the above figure, “first_cnn” refers to the deep CNN and “cnn_simple” refers to the wide CNN.

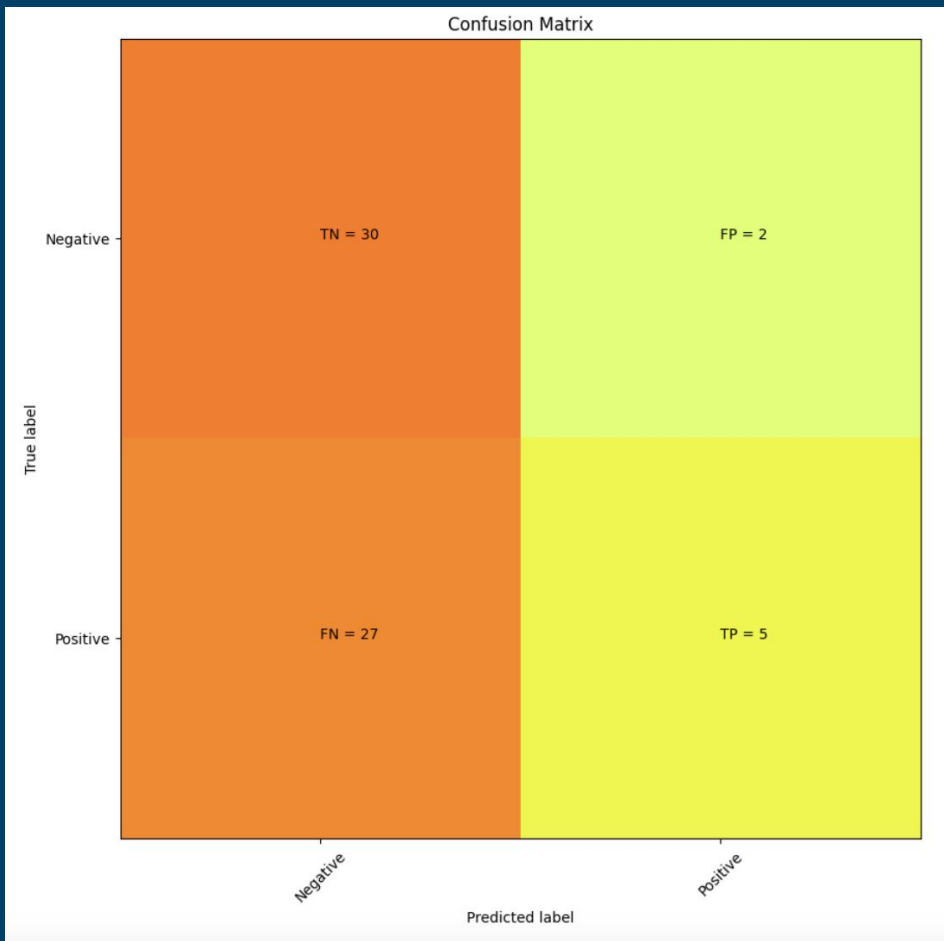
The wider model out-performs the deeper CNN model.

Results

All iterations of the ViT performed poorly on this data for every cytokine.

Here is an example of the poor performance for GM-CSF, which was easily predicted by the wide CNN model.

Tried to rescue with unscaled data and unaugmented data to no avail.



Takeaways

- ❖ It is possible to predict many cytokines from very low resolution images!
- ❖ A CNN, lacking any prior notions about what images should look like, performs far better than pre-trained ViTb32.
- ❖ The best model was the wider CNN:

```
#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'])
```

Lessons Learned

- ❖ A “fancier” model does not always lead to better performance. You can’t apply a pre-trained model to inappropriate data (no RGB cheating, either.)
- ❖ Do not assume data augmentation will improve performance, even for machine learning which favors larger datasets.

Future Work

- ❖ For each predictable cytokine, fine tune a CNN, examine learning curves, and test on data from another experiment before deploying.

Questions?

Appendix



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:

$$((\text{Conv layer height} * \text{width} * \text{channel}) + 1) * \text{units in FC layer}$$

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

$$\begin{array}{lcl} 64 * 64 * 3 & \text{---->} & 32 * 32 * 8 \quad \text{---->} \quad 64 * 1 \\ 64 * 64 * 3 & \text{---->} & 64 * 1 \end{array}$$

$$\text{Initial parameters: } (32 * 32 * 8 * 64) + (8 * (1 + (2 * 2 * 3))) = 524288 + 104 = 524,392$$

$$\text{Parameters after removing a convolutional layer: } 64 * 64 * 3 * 64 = 786,432$$

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