Predicting Perturbative Treatments from Cellular Images

Abstract:

This report investigates the effects of perturbative treatments on cellular morphology using deep learning techniques. A set of 2,687 images were acquired from Chandrasekaran et al.'s public dataset, each of which was annotated with metadata describing the treatment type applied to the cells in the image. Using the well-known cellpose library for segmentation, a two-channel dataset was created, which comprised segmented cell masks and their corresponding greyscale images. With this dataset and the corresponding metadata, the task of predicting perturbative treatments from cellular morphology is naturally framed as supervised learning problem. To solve this problem, a convolutional neural network based on the ResNet18 architecture was trained and evaluated. The model ultimately achieved training and test accuracies of 38.1% and 33%, respectively across 384 possible classes (naïve random guessing would yield 0.26% accuracy).

Introduction:

One of biology's defining themes is that the structure of living matter directly informs its function — a principle evident across scales ranging from single molecules to entire organs. The compact, spring-like structure of ATP reflects its role in storing energy for cells; the symmetrical quaternary structure of hemoglobin creates a central pocket ideal for carrying oxygen in the bloodstream; and the large, thin ears of an elephant— with their high surface area to volume ratio — are suggestive of their role in cooling the giant animals. This theme extends to cells and drugs too, and therefore, it is important to understand how various compounds affect cell morphology, for these changes to morphology can "elucidate compounds' mechanism of action or novel regulators of genetic pathways" [Chandrasekaran et al., 2024].

To this end, recent advances in microscopy and image analysis have facilitated high-throughput studies aimed at understanding cell morphology in response to genetic and chemical perturbations. These studies have yielded large datasets of labeled images, which can be used to train image-based machine learning algorithms that can ultimately generate representations of a given cells' state in response to a perturbation. For example, Chandrasekaran et al.'s 2024 paper, "Three million images and morphological profiles of cells treated with matched chemical and genetic perturbations" provides three million labeled images with the express goal of providing a "ground truth" for such tasks.

Adjacent to this problem of representation is prediction, which comprises the principal challenge discussed herein. Leveraging convolutional neural networks (CNNs), a subset of the three million images from Chandrasekaran et al.'s 2024 paper was used to train a classifier that predicts a cell's perturbation from the median aggregated image data.

Methods:

Segmentation:

As mentioned above, 2,687 images were selected from Chandrasekaran et al.'s public dataset. The specific images correspond to plate BR00116991 with 384 compound perturbations targeting specific genes. Images were provided in a "median aggregated" form, which was computed from the first three channels of the six used in the original paper.

Before training any models, images were first segmented using the well-known cellpose python library. The key output of this segmentation was a mask for each image, which indicated where biological matter was present. An example output is shown below in figures 1 and 2.

Figure 1: Example median aggregated image

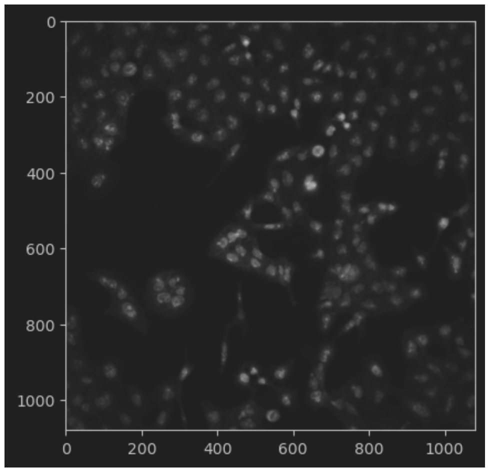
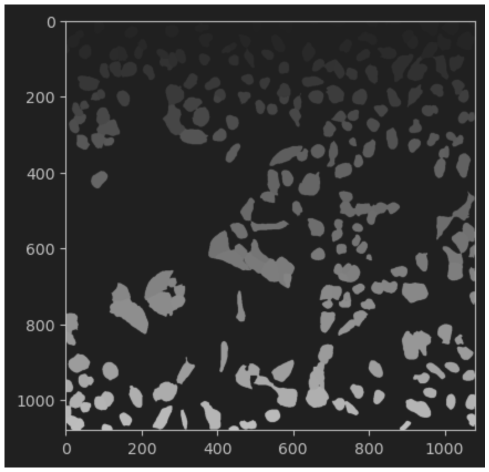


Figure 2: Mask generated from the image in figure 1



The masks for each image were saved. Taken together, each image now comprised two "channels": one for the underlying pixels, and one for the segmented mask. These channels were to be taken as input for the classification. The explicit supervised machine learning problem is now evident: predict the perturbation from this two-channel image.

General Outline of Techniques:

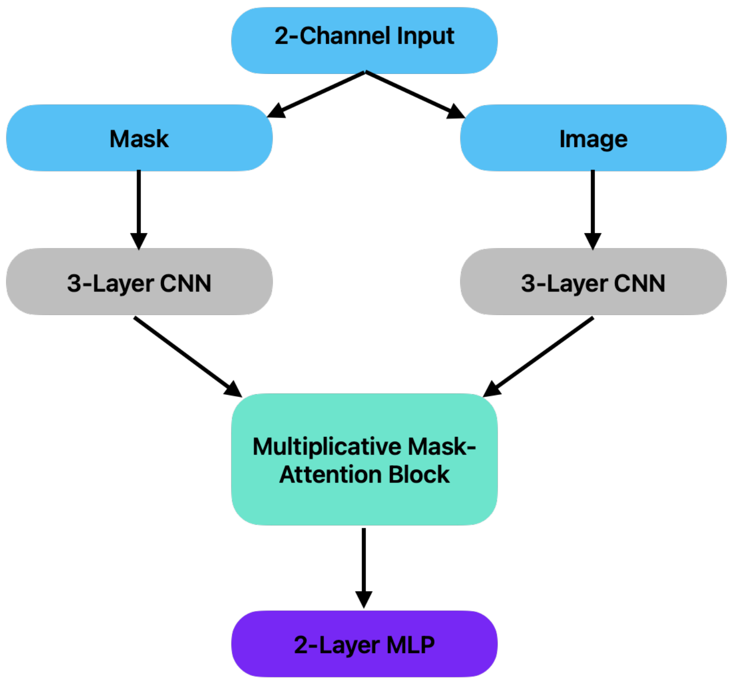
Though dozens of classifiers were ultimately explored, they all generally followed one of two approaches:

1. Directly train a CNN on the two-channel masked image data. The CNN's final layer is a 384-dimensional vector, with each entry corresponding to the network's probability that the given image was generated by one of the 384 perturbations.
2. Use a pre-trained deep-learning model to generate a representation of the data in some high-dimensional feature space. Then, use classical machine learning techniques to fit a classifier on the data.

CNN Architecture:

Though several architectures were explored, the most promising was a custom CNN (implemented in PyTorch). The model architecture processes the mask and the image separately in their own simple 3-layer CNN with batch normalization and max-pooling. These results are then combined via a multiplicative "attention" mechanism, which applies additional convolutions and activations to the mask before multiplying it with the image. The inspiration of this architecture is the physical fact that the mask multiplied by the image yields an image with a higher signal to noise ratio. The resulting image is passed through a 2D adaptive average pooling layer and is then fed to a two-layer feedforward neural network with dropout for the final classification. A block diagram of the model is shown in figure 3 below.

Figure 3: Main CNN architecture



The model was trained with the following parameters:

- Loss function: Cross-entropy loss with label smoothing (0.1)
- Optimizer: Adam Optimizer with weight decay of 0.0001
- Learning rate scheduler: One-Cycle LR (max 0.005)
- Number of epochs: 200 scheduled (160 executed due to compute constraints)
- Hardware: MacOS / AMD Radeon Pro 5300M
- Total training time: ~10 hours

Pre-Trained Embeddings:

The other principal method used was to use a pre-trained model to generate a high-dimensional representation of the underlying data. Fine-tuning of these models on the given dataset was attempted but proved infeasible given hardware limitations. A few of the models used include: ResNet18, ResNet50, and EfficientNet_B0. The general approach was to download these models and truncate the forward pass before the final layer. The resulting activations of the penultimate layer were used as features for classical algorithms. This also enabled exploration of the feature space via UMAP, and t-SNE.

Compound models were also explored whereby pre-trained models were used as building blocks for more complex models with additional layers. These models were trained in two phases. In the first phase, the weights of the pre-trained model were frozen while the extra layers were trained. In the second phase, all weights were unfrozen.

Code Availability:

The complete pipeline for image segmentation and model training is available at https://github.com/wvietor/stat4243_project2. The segmentation code is available in "segmentation.ipynb". The principal neural network is available in "main_attention.ipynb". The pre-trained embedding code is available in "main_embed.ipynb" and "main_resnet50_error_with_valid_plots.ipynb" (note that this notebook is considered deprecated and is thus non-functional; it does contain performance details for resnet-50 though). Code to generate graphs not seen in other notebooks is available in "perf_graphs.ipynb." Other exploratory models are available in the Jupyter notebooks in the "deprecated" folder.

Results:

Neural Network:

The custom CNN shown in figure 3 seemed to show promise (achieving a training accuracy of 0.381 and a test accuracy of 0.33 as shown in table 1) but needs significant refinement and tuning to achieve better performance. The gap between training and test performance is narrow, which suggests the model is not overfitting within 160 epochs, and thus implies that additional training could likely improve results. Additionally, the training accuracy and loss appear to be improving at an appreciable rate even in later epochs, which further suggests that more training could improve results. However, despite the steady trend in the training loss and accuracy, the overall rate of improvement is still quite low, and could potentially be improved by a better architecture. Nevertheless, the observed trend for the CNN architecture described in figure 3 was the best of the CNN architectures that were explored (though notably on far fewer epochs due to compute constraints).

Perhaps the most notable feature of the training and test performance graphs shown in figures 4 and 5 is the apparent divergence and reconvergence of the test performance. It is difficult to fully explain these divergences, though visual inspection of figures 4, 5, and 6 suggests that the comparatively higher learning rate used between epochs ~40 and ~130 could be one cause of this noise.

Table 1: Performance of custom-architected CNN after 160 epochs

Classifier	Training Accuracy	Test Accuracy

Custom CNN (see figure 3)	0.381	0.33

Figure 4: Cross Entropy Loss vs. Epoch

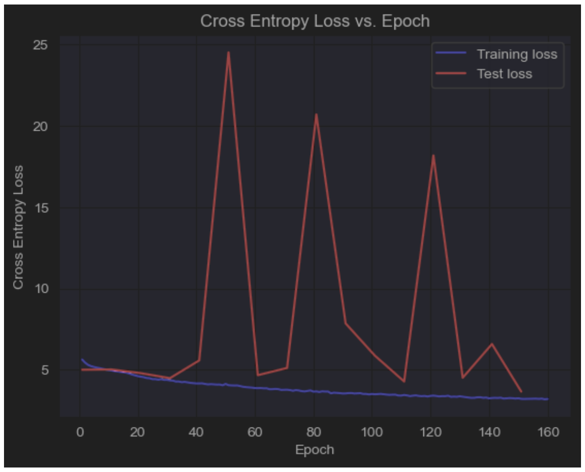


Figure 5: Accuracy vs. Epoch

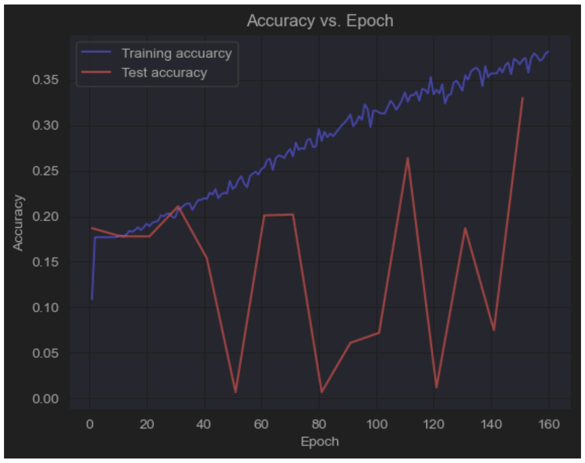
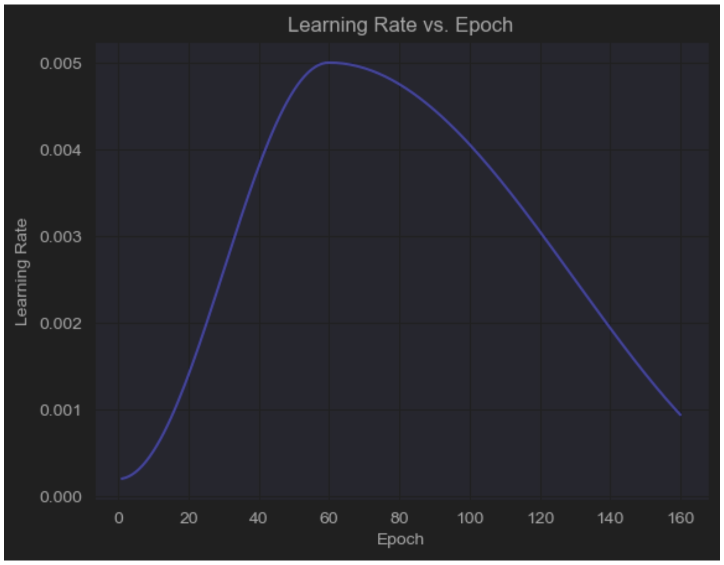


Figure 6: Learning Rate vs. Epoch



Pre-Trained Embeddings:

The pre-trained embeddings approach generally yielded weak performance. Of the pre-trained embedding classifiers, the best test overall accuracy of 0.2848 was achieved by a Ridge classifier on the ResNet-50 model's features. Notably this test accuracy was substantially lower than the training accuracy of 0.8248, which suggests substantial overfitting. Increasing the ridge parameter did not yield a higher test accuracy. This trend persisted for the EfficientNet_B0 features, with a random forest achieving perfect training accuracy but paltry test accuracy of just 0.1815 as shown in table 2. It's possible that fine tuning could improve the performance of this approach, though in general, this method yielded minimal results.

Table 2: Performance of Classical ML Models with Efficient Net_B0 Features

Classifier	**Training Accuracy**	**Test Accuracy**
Random forest	1	0.1815
Bernoulli naïve-bayes	0.3056	0.1640

Figure 7: UMAP Embeddings for EfficientNet_B0

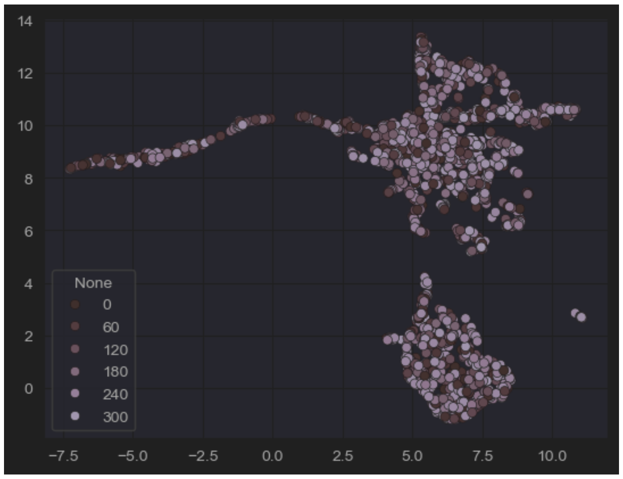
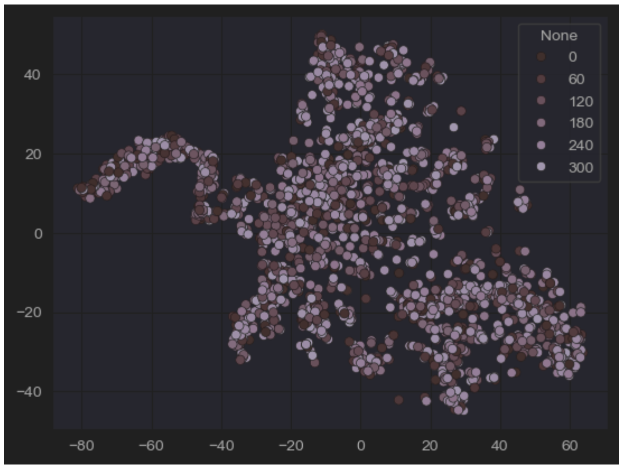


Figure 8: t-SNE Embeddings for EfficientNet_B0



Pre-Trained Embeddings (ResNet-50):

Classifier	**Training Accuracy**	**Test Accuracy**
Ridge classifier	0.8248	0.2838

Conclusion:

Thus, the analysis techniques described above were able to predict the perturbative compound with limited success. CNNs appear to be a sound approach to the problem, with the architecture described in figure 3 achieving solid results given the large size of the target class (384 possible compounds). As shown in the analysis section, the model seems to have the capacity for accurate prediction without excessively overfitting to the training set. Nevertheless, significant optimizations are necessary to push the accuracy to higher values. Some of these optimizations may be attainable though a better learning rate schedule or improved hyperparameter values, though it's likely that novel architectures and/or substantially more training time with vastly more computational resources will be necessary to achieve accuracy exceeding 80% or 90%.

Works Cited

Chandrasekaran, S.N., Cimini, B.A., Goodale, A. *et al.* Three million images and morphological profiles of cells treated with matched chemical and genetic perturbations. *Nat Methods* **21**, 1114–1121 (2024). https://doi.org/10.1038/s41592-024-02241-6.