Deep Learning for Drug Response

Final Presentation
4/17/2025

Katya Aukamp, Ameya Chander, Luc Rieffel

207 Final Project

Our Data

RxRx2

131,953 fluorescence microscopy images of cells exposed to different drug treatments. Each row is an image, with descriptors of the image. These embeddings can be used for classification, clustering or feature importance analysis

Images

These are 128-dimensional features, they are not directly human-interpretable.

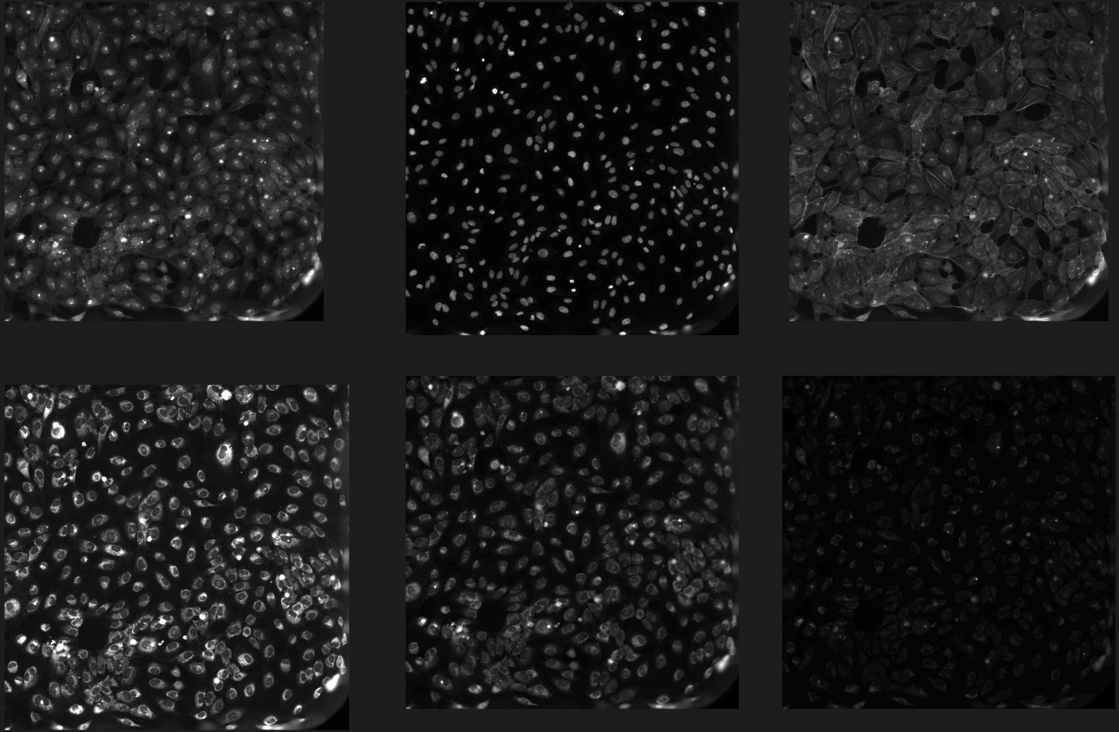
Features

These 128 features are numeric representations of the microscopy images, they are not raw pixels but high level extracted patterns describing cell morphology, stain intensity and structural changes

Treatments

There are 434 unique treatments with 288 samples each

Rxrx2 Images



Why this matters

Biological Relevance

Understanding how different drugs affect cell behavior is critical to drug discovery and disease treatment

Biomedical Research

Help researchers analyze microscopy images with deep learning to extract meaningful features

Real World Impact

Be able to identify new drug effects faster, and understand what happens on a cell level when treatment is administered

Narrowing the scope

Interferons

- Are natural occurring proteins produced by the body to help fight infection and other diseases
- 4 treatments at different concentrations:
  - **IFN-alpha-A:** Immune regulatory protein that enhances cell-mediated cytotoxicity against tumor cells
  - **IFN-beta:** Works to reduce inflammation and protect nerve cells
  - **IFN-gamma:** Plays a role in regulating antibody production and immune cell development
  - **IFN- omega:** Inhibits proliferation of a variety of tumor cell lines in vitro

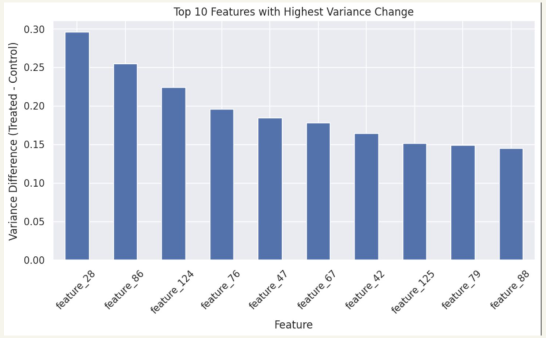
The Focused Sub-Groups

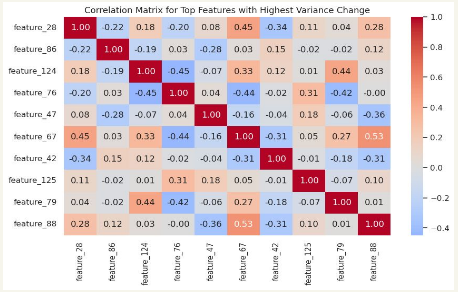
Treatment	Samples Count	Downsamples
IFN- alpha	288	**287**
IFN-beta	287	287
IFN-gamma	287	287
IFN-omega	288	287
Control	7579	287

Initial Data Cleaning

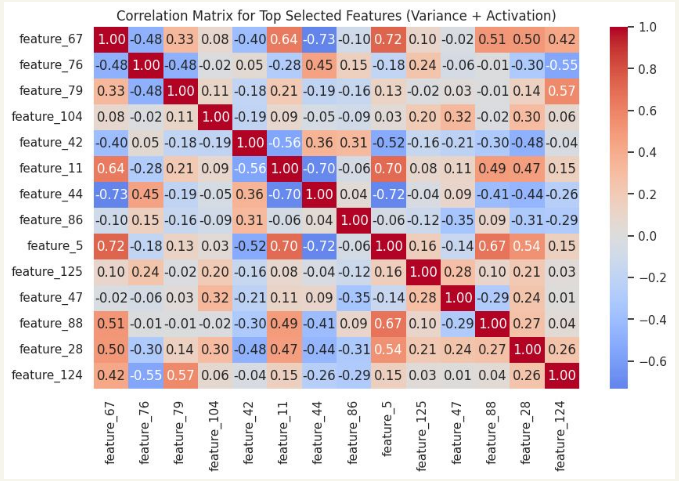
- Downloaded the two parts of the data > Image Embeddings and Image Metadata
- The embedded image features are numerical representations of cell images generated by a deep learning model. Instead of analyzing raw pixels, the model extracts 128 key features that summarize cell structure, stain intensity, and morphological changes, allowing us to compare how different treatments affect cells.
- Metadata gives information on the well and treatment administered
- Of the data there are 7,579 samples that have no treatment
- With our focus on interferons our ultimate dataset since is 1,435

Summary Statistics

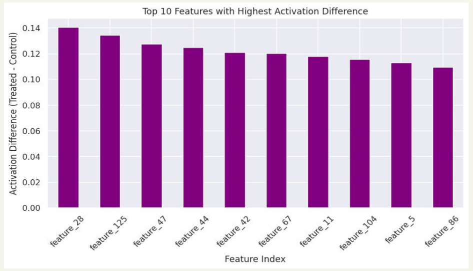


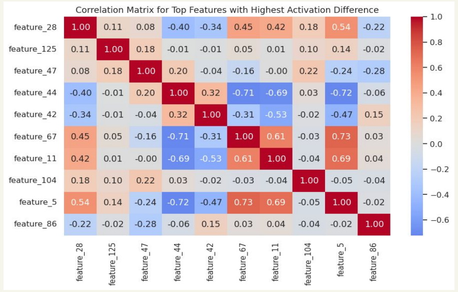


	feature_28	feature_86	feature_124	feature_76	feature_47	feature_67	feature_42	feature_125	feature_79	feature_88
feature_28	1.00	-0.22	0.18	-0.20	0.08	0.45	-0.34	0.11	0.04	0.28
feature_86	-0.22	1.00	-0.19	0.03	-0.28	0.03	0.15	-0.02	-0.02	0.12
feature_124	0.18	-0.19	1.00	-0.45	-0.07	0.33	0.12	0.01	0.44	0.03
feature_76	-0.20	0.03	-0.45	1.00	0.04	-0.44	-0.02	0.31	-0.42	-0.00
feature_47	0.08	-0.28	-0.07	0.04	1.00	-0.16	-0.04	0.18	-0.06	-0.36
feature_67	0.45	0.03	0.33	-0.44	-0.16	1.00	-0.31	0.05	0.27	0.53
feature_42	-0.34	0.15	0.12	-0.02	-0.04	-0.31	1.00	-0.01	-0.18	-0.31
feature_125	0.11	-0.02	0.01	0.31	0.18	0.05	-0.01	1.00	-0.07	0.10
feature_79	0.04	-0.02	0.44	-0.42	-0.06	0.27	-0.18	-0.07	1.00	0.01
feature_88	0.28	0.12	0.03	-0.00	-0.36	0.53	-0.31	0.10	0.01	1.00

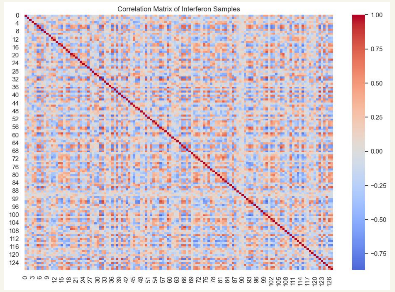


	feature_67	feature_76	feature_79	feature_104	feature_42	feature_11	feature_44	feature_86	feature_5	feature_125	feature_47	feature_88	feature_28	feature_124
feature_67	1.00	-0.48	0.33	0.08	-0.40	0.64	-0.73	-0.10	0.72	0.10	-0.02	0.51	0.50	0.42
feature_76	-0.48	1.00	-0.48	-0.02	0.05	-0.28	0.45	0.15	-0.18	0.24	-0.06	-0.01	-0.30	-0.55
feature_79	0.33	-0.48	1.00	0.11	-0.18	0.21	-0.19	-0.16	0.13	-0.02	0.03	-0.01	0.14	0.57
feature_104	0.08	-0.02	0.11	1.00	-0.19	0.09	-0.05	-0.09	0.03	0.20	0.32	-0.02	0.30	0.06
feature_42	-0.40	0.05	-0.18	-0.19	1.00	-0.56	0.36	0.31	-0.52	-0.16	-0.21	-0.30	-0.48	-0.04
feature_11	0.64	-0.28	0.21	0.09	-0.56	1.00	-0.70	-0.06	0.70	0.08	0.11	0.49	0.47	0.15
feature_44	-0.73	0.45	-0.19	-0.05	0.36	-0.70	1.00	0.04	-0.72	-0.04	0.09	-0.41	-0.44	-0.26
feature_86	-0.10	0.15	-0.16	-0.09	0.31	-0.06	0.04	1.00	-0.06	-0.12	-0.35	0.09	-0.31	-0.29
feature_5	0.72	-0.18	0.13	0.03	-0.52	0.70	-0.72	-0.06	1.00	0.16	-0.14	0.67	0.54	0.15
feature_125	0.10	0.24	-0.02	0.20	-0.16	0.08	-0.04	-0.12	0.16	1.00	0.28	0.10	0.21	0.03
feature_47	-0.02	-0.06	0.03	0.32	-0.21	0.11	0.09	-0.35	-0.14	0.28	1.00	-0.29	0.24	0.01
feature_88	0.51	-0.01	-0.01	-0.02	-0.30	0.49	-0.41	0.09	0.67	0.10	-0.29	1.00	0.27	0.04
feature_28	0.50	-0.30	0.14	0.30	-0.48	0.47	-0.44	-0.31	0.54	0.21	0.24	0.27	1.00	0.26
feature_124	0.42	-0.55	0.57	0.06	-0.04	0.15	-0.26	-0.29	0.15	0.03	0.01	0.04	0.26	1.00





	feature_28	feature_125	feature_47	feature_44	feature_42	feature_67	feature_11	feature_104	feature_5	feature_86
feature_28	1.00	0.11	0.08	-0.40	-0.34	0.45	0.42	0.18	0.54	-0.22
feature_125	0.11	1.00	0.18	-0.01	-0.01	0.05	0.01	0.10	0.14	-0.02
feature_47	0.08	0.18	1.00	0.20	-0.04	-0.16	-0.00	0.22	-0.24	-0.28
feature_44	-0.40	-0.01	0.20	1.00	0.32	-0.71	-0.69	0.03	-0.72	-0.06
feature_42	-0.34	-0.01	-0.04	0.32	1.00	-0.31	-0.53	-0.02	-0.47	0.15
feature_67	0.45	0.05	-0.16	-0.71	-0.31	1.00	0.61	-0.03	0.73	0.03
feature_11	0.42	0.01	-0.00	-0.69	-0.53	0.61	1.00	-0.04	0.69	0.04
feature_104	0.18	0.10	0.22	0.03	-0.02	-0.03	-0.04	1.00	-0.05	-0.04
feature_5	0.54	0.14	-0.24	-0.72	-0.47	0.73	0.69	-0.05	1.00	-0.02
feature_86	-0.22	-0.02	-0.28	-0.06	0.15	0.03	0.04	-0.04	-0.02	1.00



Our Approach

- Our first approach to work at a subcategory level, where we group treatments into larger groups, with this approach we tried:
    - Classify images with CNN into treatment category - generally this did not work and we had very low results even though our summary statistics showed differences in the data
    - To try and get better image classification we tried a Vision Transformer this while promising from the literature our images and GPU were too small to be able to work
- Ultimately the workflow that works was to narrow the topic to a specific treatment subgroup (Interferons) and look at the specific treatments use, generate our own embeddings and then classify the images based on those embeddings

Overview of the Workflow

1. **Build SimCLR Encoder**
   Why: *Learn rich, stain-invariant features from unballed images*

2. **Train with Contrastive Loss**
   Why *: Encourage model to cluster similar views & separate unrelated ones*

3. **Freeze Encoder**
   Why*: Preserve the learned representations*

4. **Train Classifier Head**
   Why*: Fine-tune on labeled data to make interpretable predictions*

5. **Extract Embeddings**
   Why*: Prepare inputs for other classifier/visualizations*

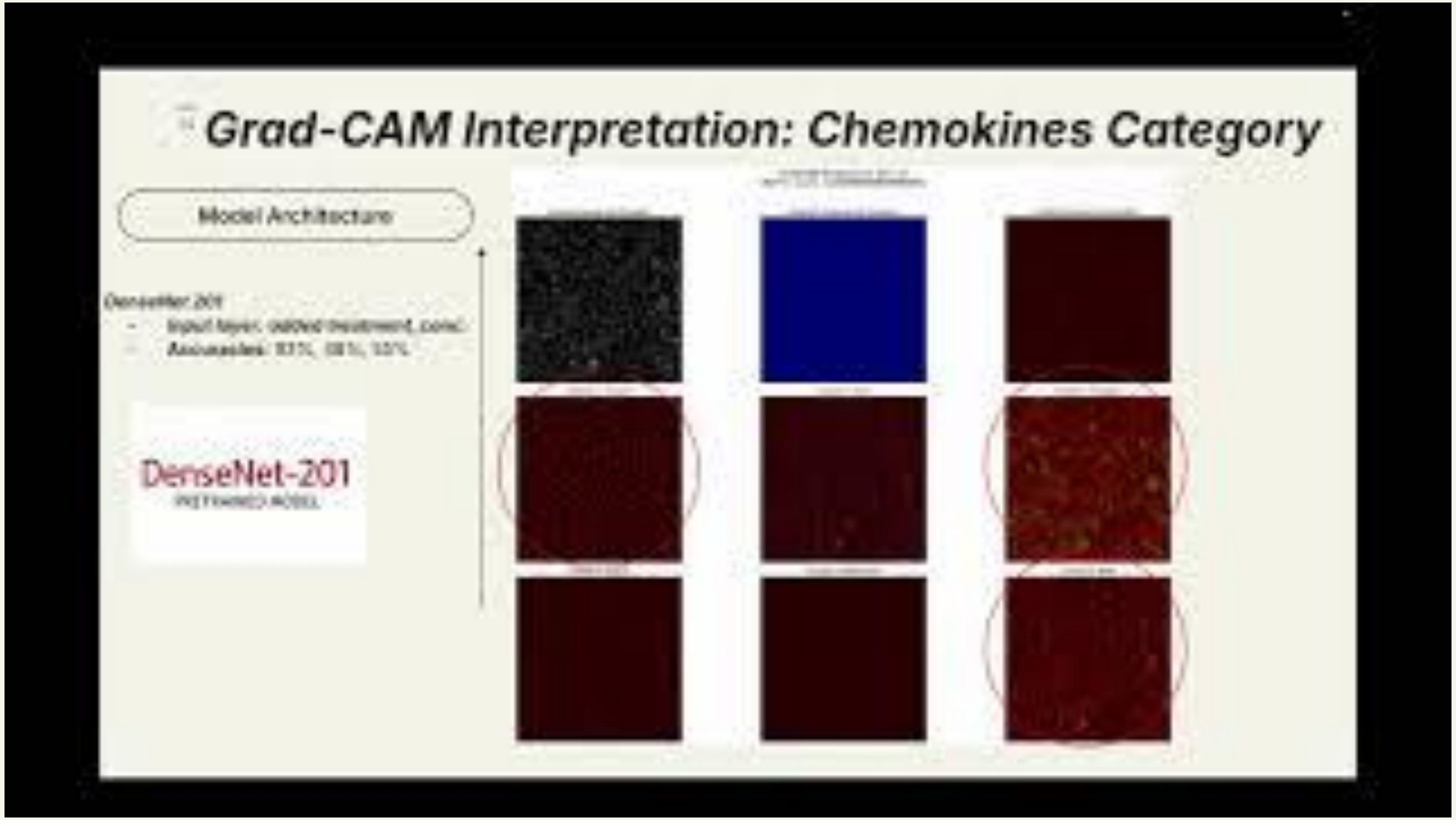
6. **Train Random Forest**
   Why*: Compare how well we can classify images by the features extracted*

7. **Run Grad-CAM**
   Why*: Understand model focus areas across the 6 stain chanels*

Optimizations Made

Before	After
Had shallow projection head with minimal transformation	Added a deeper projection head with two dense layers, batch normalization, and dropout
Had shallow projection head with minimal transformation	Build a full `classifier_model` by adding a classification head to `simclr_model`
Loaded one image at a time	Pre-processed the images so that the model could run in under 30 minutes from 2 hours

Luc Presentation Video



Grad-CAM Interpretation: Chemokines Category

Model Architecture

DenseNet 201
- Input layer: added treatment, conc.
- Accuracies: 93%, 88%, 53%

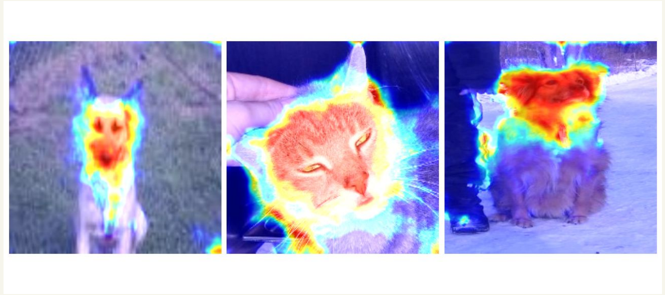
CNN Interpretation: GRADCAM

GRADCAM: (gradient-weighted class activation mapping)

- Explainability method for CNNs
- Highlights the regions of an input image that are most important for classification
- Uses gradients in the last convolutional layer to produce a mapping of important features/regions

GRADCAM Interpretation:

- Warm, red, or highlighted regions indicate areas of the image the model is activating on/responding to

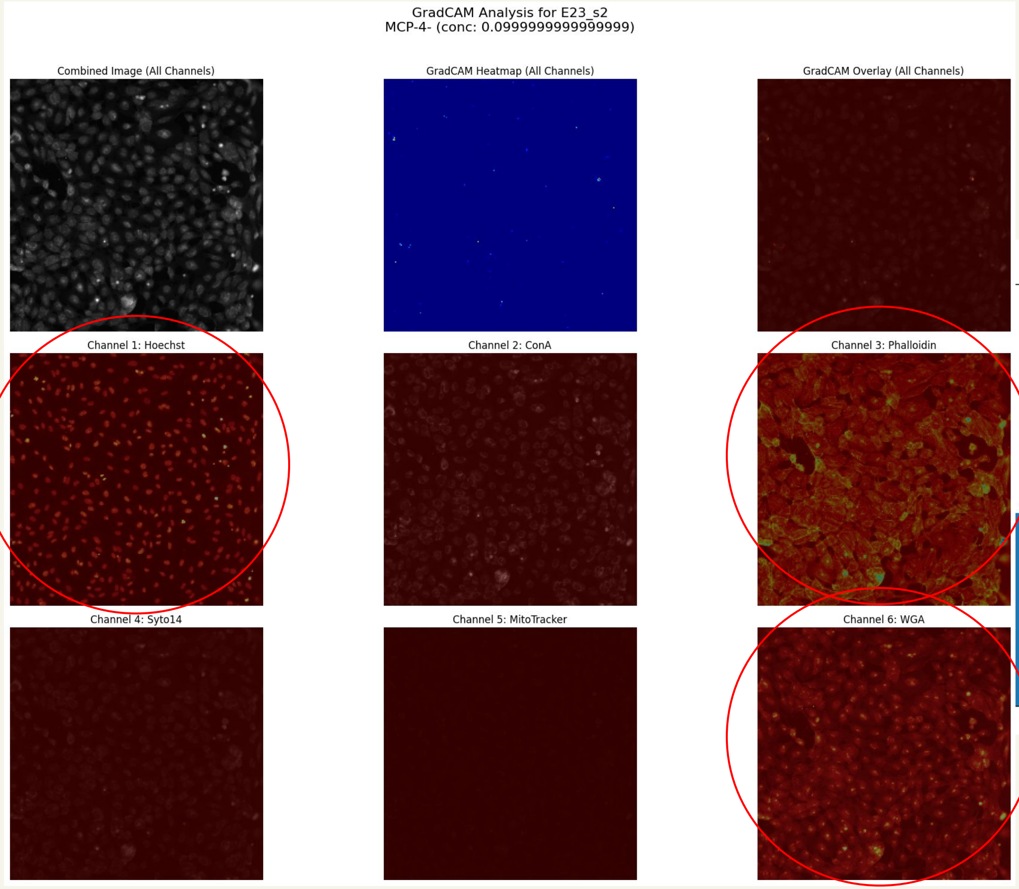


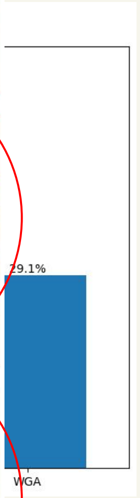
Grad-CAM Interpretation: Chemokines Category

Model Architecture

DenseNet 201

- ***Input layer: added treatment_conc:***
- **Accuracies:** 93%, 48%, 55%


GradCAM Analysis for E23_s2
MCP-4- (conc: 0.0999999999999999)
Combined Image (All Channels)
GradCAM Heatmap (All Channels)
GradCAM Overlay (All Channels)
Channel 1: Hoechst
Channel 2: ConA
Channel 3: Phalloidin
Channel 4: Syto14
Channel 5: MitoTracker
Channel 6: WGA



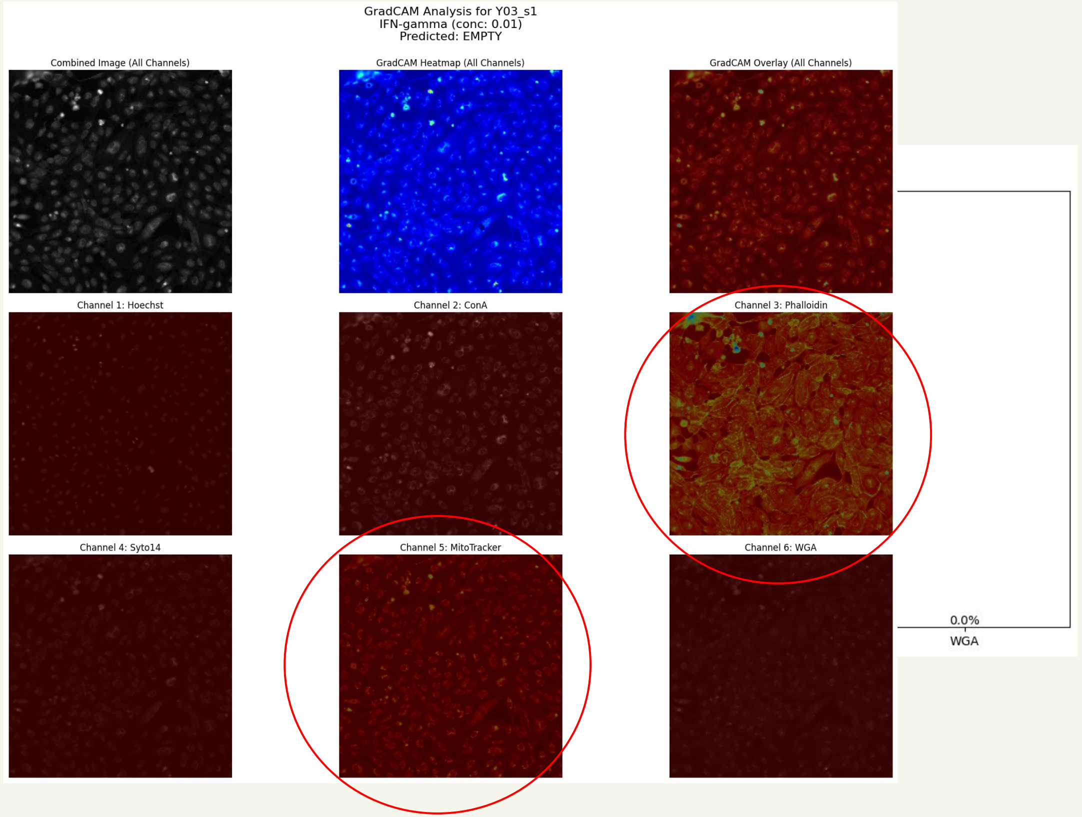
Grad-CAM Interpretation: Interferons Category

Model Architecture

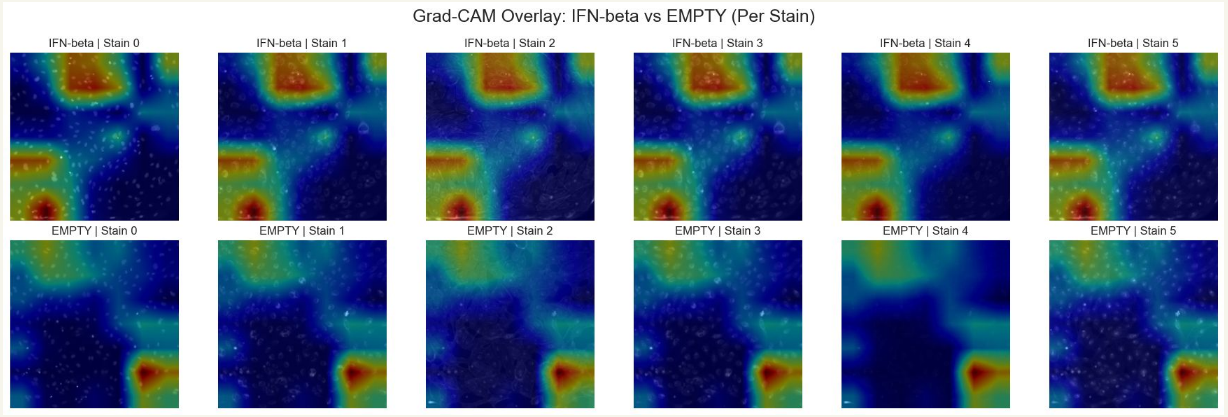
DenseNet 201

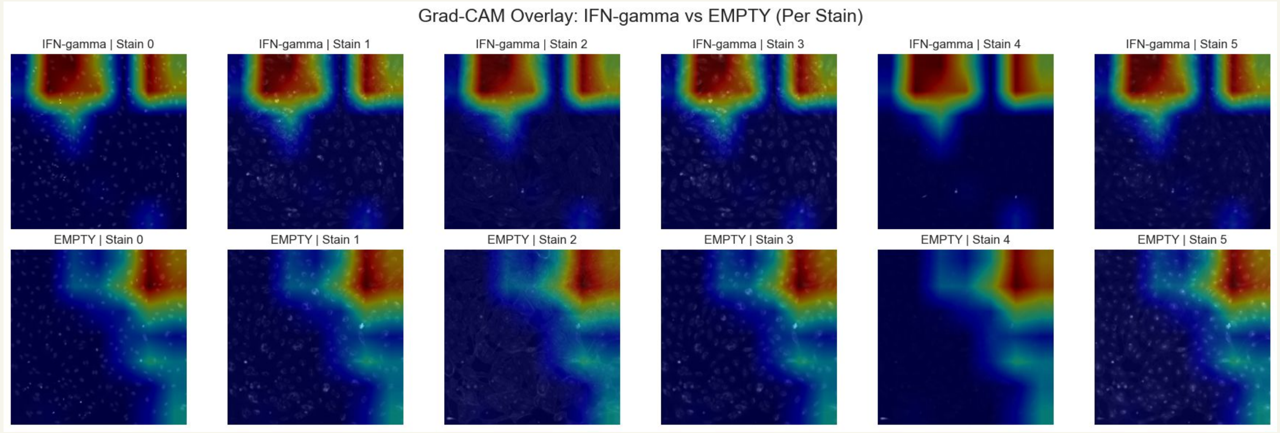
- ***Input layer: added treatment_conc:***
- **Accuracies:** 95%, 44%, 53%

DenseNet-201
PRETRAINED MODEL

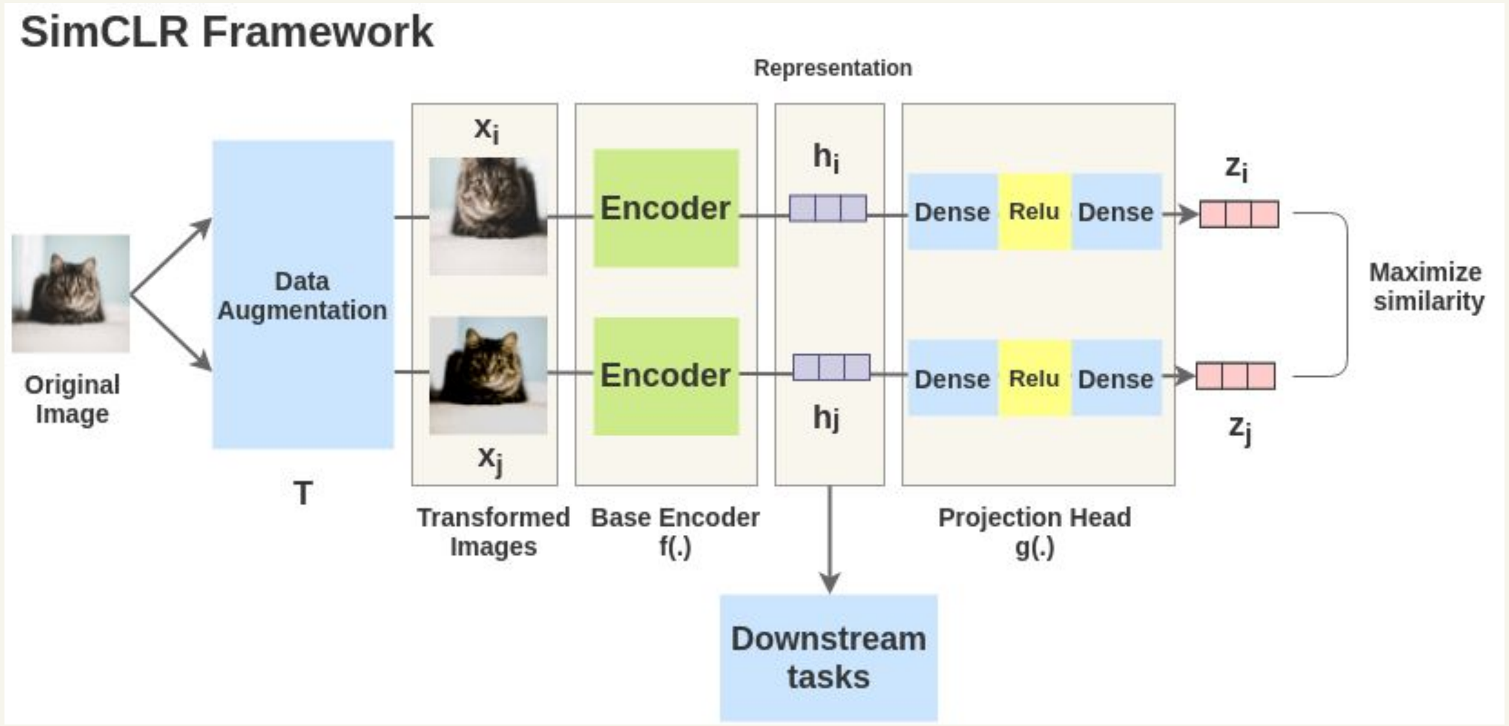


Grad-CAM Interpretation: IFN-beta





1. SimCLR Contrastive Learning



Model Pre-processing

- Downsampled to 287 samples per treatment
- Loaded all 6 stain images for each sample
- Resizes and normalizes them
- Stacks them into a single 6-channel tensor
- Saves the result as a .npy file for faster future loading
- This was done once and the .npy files stored so we didn’t have to run again

Image Augmentation

- Because our model works by comparing images we need to generate a copy of the cell with augmentations
- We did the following augmentations:
    - Brightened All Images
    - Random Flip
    - Random crop + resize
    - Random brightness and crop
    - Gaussian noise to simulate camera/staining variation

Model Loss Function: SimCLR

- NT-Xent (Normalized Temperature-scaled Cross Entropy) is a contrastive loss used in SimCLR to train the model without labels.

- It works by comparing two augmented views of each image (positive pairs) and learning to **maximize their similarity** while pushing away **all other views (negatives)** in the batch.

- L2 normalization rescales vectors to the same length so we compare **what** an image shows, not **how strongly** it shows it.

- A **similarity matrix** of all pairs is constructed using matrix multiplication:

- The **temperature parameter** (τ = 0.5 here) controls sharpness of softmax:
  - Lower τ (e.g., 0.1) focuses on strongest matches
  - Higher τ (e.g., 1.0) smooths similarity scores

- .The loss function uses **sparse categorical cross-entropy**, where each embedding should most closely match **its positive pair**

- The final value is the **mean loss over the batch**, encouraging positive pairs to align and negatives to separate in the embedding space.

Building the SimCLR Feature Extractor

- Input: 6-channel cell microscopy images

- Use DenseNet121 pre-trained on ImageNet as the backbone

- Images were projected from 6-channels -> channels via Conv2D layers

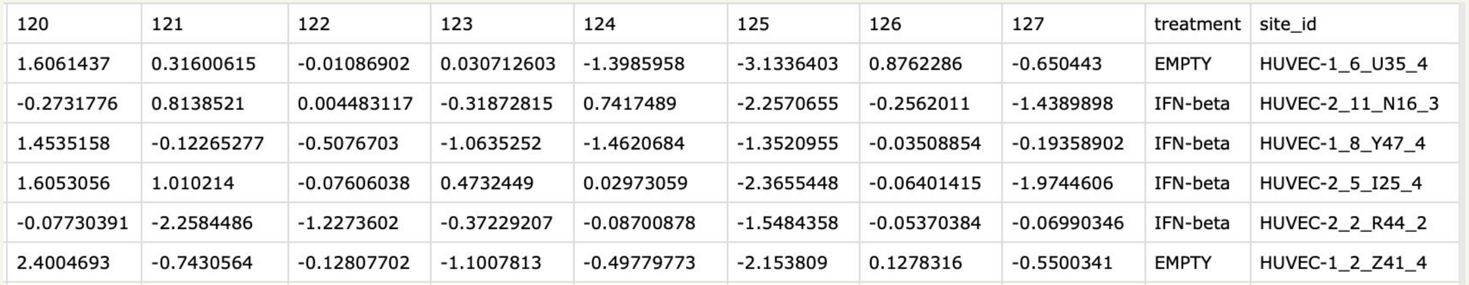
- Each training step:
    - Augment the image twice
    - Pass both through the encoder
    - Generate two embeddings: z_i & z_j

- NT-Xent loss to group similar images and push away different images

- Trained for 20 epochs using Adam optimizer, with learning rate = 0.0004

SimCLR Outcome

Treatment	Loss
IFN- alpha	2.84
IFN-beta	2.86
IFN-gamma	2.83
IFN-omega	2.84



120	121	122	123	124	125	126	127	treatment	site_id
1.6061437	0.31600615	-0.01086902	0.030712603	-1.3985958	-3.1336403	0.8762286	-0.650443	EMPTY	HUVEC-1_6_U35_4
-0.2731776	0.8138521	0.004483117	-0.31872815	0.7417489	-2.2570655	-0.2562011	-1.4389898	IFN-beta	HUVEC-2_11_N16_3
1.4535158	-0.12265277	-0.5076703	-1.0635252	-1.4620684	-1.3520955	-0.03508854	-0.19358902	IFN-beta	HUVEC-1_8_Y47_4
1.6053056	1.010214	-0.07606038	0.4732449	0.02973059	-2.3655448	-0.06401415	-1.9744606	IFN-beta	HUVEC-2_5_I25_4
-0.07730391	-2.2584486	-1.2273602	-0.37229207	-0.08700878	-1.5484358	-0.05370384	-0.06990346	IFN-beta	HUVEC-2_2_R44_2
2.4004693	-0.7430564	-0.12807702	-1.1007813	-0.49779773	-2.153809	0.1278316	-0.5500341	EMPTY	HUVEC-1_2_Z41_4

Training the Classification Model

- A new model was built to train a classifier on **learned features using Random Forest Classifier**
  - **Input**: SimCLR encoder
  - **Output:** Dense layers ending in a sigmoid for binary classification

- Trained on labeled data: Control vs Treatment

- Evaluated with accuracy, precision, recall, and F1 score

- The model uses **100 decision trees**, and we set a random seed to keep results consistent each time it runs.

- This gave us a clear sense on how well the SimCLR embeddings could be used for actual classifications

Classifier Outcome

Treatment	Validation Accuracy	Test Accuracy	Accuracy	Precision	Recall	F1
IFN- alpha	54.78%	53.91%	53.91%	54.45%	54.39%	53.91%
IFN-beta	62.61%	69.57%	68.70%	67.80%	70.18%	68.97%
IFN-gamma	51.30%	52.17%	62.61%	61.29%	66.67%	63.87%
IFN-omega	67.83%	56.52%	51.30%	50.82%	54.39%	52.54%

Impacts & Uses

- We used SimCLR to teach our model what cells “look like “ – without needing labels – by comparing different augmented views

- Then we built classifiers on top of those learned features to spot differences between control cells and treated ones

- **Speed-run drug discovery**: Spot cellular changes from new compounds *before* we even know what they do.

- **Unlock hidden phenotypes**: Reveal subtle treatment effects that humans (and even supervised models) might miss.

- **Drug repurposing made smarter**: Find unexpected similarities between treatments by clustering learned cell embeddings.

References

- Sypetkowski et al. 2023
- Cuccarese, M. F., Earnshaw, B. A., Heiser, K., Fogelson, B., Davis, C. T., McLean, P. F., Gordon, H. B., Skelly, K.-R., Weathersby, F. L., Rodic, V., Quigley, I. K., Pastuzyn, E. D., Mendivil, B. M., Lazar, N. H., Brooks, C. A., Carpenter, J., Jacobson, P., Glazier, S. W., Ford, J., ... Gibson, C. C. (2020). Functional immune mapping with deep-learning enabled phenomics applied to immunomodulatory and COVID-19 drug discovery. bioRxiv. https://doi.org/10.1101/2020.08.02.233064
- Minhaz, M. (2020, July 28). *SimCLR explained in simple terms*. Medium. https://medium.com/one-minute-machine-learning/simclr-explained-in-simple-terms-3fa69af45ff9
- Scaler. (n.d.). *Self-supervised learning in Keras*. Scaler Topics. https://www.scaler.com/topics/keras/self-supervised-learning-keras/
- Sharma, A. (2021, June 23). *NT-Xent (Normalized Temperature-scaled Cross Entropy) loss — explained and implemented in PyTorch*. Towards Data Science. https://towardsdatascience.com/nt-xent-normalized-temperature-scaled-cross-entropy-loss-explained-and-implemented-in-pytorch-cc081f69848/