What is the name of your organization?
NYU School of Medicine, Applied Bioinformatics Laboratories
What is the best email address to contact you?
aristotelis.tsirigos@nyumc.org
What is the project?
We have recently used Google's Inception V3 to:

- automatically classify lung cancer images into lung cancer subtypes (AUC ~0.97, better that pathologists!)

- predict gene mutations using only images

We want to:

- expand our study in breast, prostate and skin cancers

- optimize the neural network architecture for faster training

- visualize the features learned by the network
What does the data look like?
Images, labeled by cancer subtype and gene mutations.
What is the proposed scope of the projects?
Minimum number = 2:

- 1 person should be able to work on one new cancer type (e.g. melanoma)

- Another person will be necessary so that new network architectures can be explored

If 3-4 students are interested, we can focus on other cancer types. Data exists for at least 5 cancer types.
What are the rubrics of success?
Outperform doctors in classifying tumor images. Faster training of neural networks.
What is the relevant background needed with respect to the project?
Machine learning, neural networks.
What is the relevant organizational, project or institutional history?
NYU Medical Center is a leading healthcare provider and one of the top Medical Schools in research. The Applied Bioinformatics Laboratories has engaged in collaborative research with "wet bench" investigators at NYU, with multiple publications in top ranked scientific journals.
How will the organization support and mentor the students?
Share our computational workflows, frequently meet to discuss progress.
Additional Feedback?

Efficient cancer classification from tissue slide images using CNNs

DS-GA 1006: Capstone Project Memo

Joyce Wu (jmw784)
Eduardo Fierro (eff254)
Raul Delgado Sanchez (rds491)

November 9, 2017

1 Introduction

Deep convolutional neural networks (CNNs) have achieved state-of-the-art performance on many image classification tasks. We aim to apply the latest developments in CNNs on the task of cancer classification using labeled tissue slide images. Previous work has been done by our advisors Nicolas Coudray, Narges Razavian, and Aristotelis Tsirigos on this task. Their implementation of Google's Inception V3, a deep ResNet, obtained an impressive AUC of 0.97 - better than pathologists. Our goal is to replicate their work with a modified, simpler architecture that would be faster for training and prediction.

2 Data

We downloaded 1983 high-resolution tissue slide images of cancerous and non-cancerous tissue from Genomic Data Commons Database [1]. For lung cancer, this includes:

1. 609 slides with LUSC (Lung Squamous Cell Carcinoma)
2. 567 slides with LUAD (Lung Adenocarcinoma)

3. 459 slides with normal tissue

We are also working with 2370 breast and kidney tissue slide images (same source).

3 Methodology

3.1 Data pre-processing

1. **Image tiling:** As the original tissue slide images are high resolution (20k), we tiled the original images into smaller images for the neural network to process. For tiling, we used the open source package `openslide`. The images were tiled into 512 $\times$ 512 pixels tiles. Tiles with more than 50% of background (white space) were removed.
2. **Train / validation / test set split:** To avoid leakage, we ensured that tiles that came from the same patient were included in the same dataset (train, valid, or test).
3. **Parsing and concatenating metadata:** Alongside the images, we also had clinical data about the patient, including but not limited to: age at diagnosis, the number of cigarettes smoked per day, and gender. Furthermore, we had the x, y position of the tile from the original image. To allow the neural network to take advantage of this information, we encoded them into numbers (replaced null values with either the mean or 0 and one hot encoded the categorical variables), reshaped them into the size of the image, and then concatenated them to the image as extra channels.

3.2 Data augmentation

Using the `Tranforms` library from `Torchvision`, along with self-written functions and classes using the Python `Pillow` library, we augmented the training data by randomly introducing positional transforms such as: a vertical flip, a horizontal flip, or a rotation of 0^o, 90^o, 180^o or 270^o degrees. Additionally, we randomly adjusted the hue, gamma, brightness, contrast, and saturation of the image.

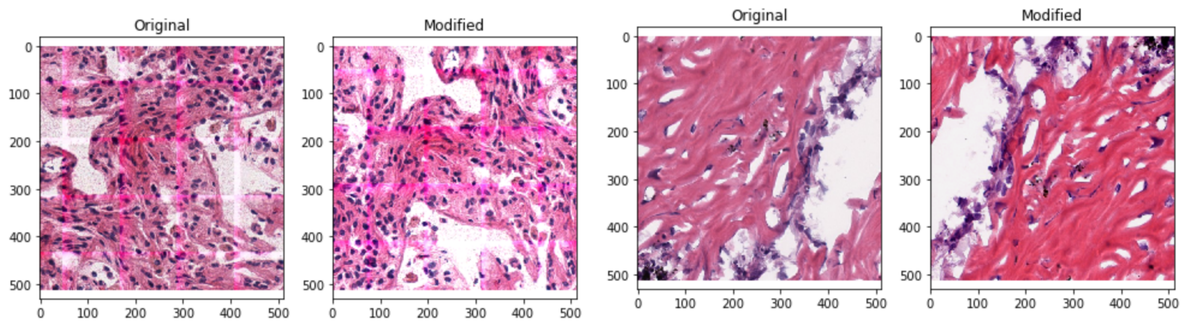

Figure 1: Data Augmentation Example

3.3 Weight initialization

As deep neural networks are highly non-convex, it's important to have a proper initialization. While small initial weights lead to signal vanishing through the layers, large weights can make the gradient explode. Therefore, we tried the following initialization methods.

1. **Gaussian:** Standard method for initializing neural networks, which fills the weights vector with random Gaussian noise centered at 0.
2. **Xavier:** Xavier initialization samples weights from a uniform distribution centered around zero and bounded by a function of the layer size. This initialization maintains variances of the activations and back-propagated gradients through the network, preventing both vanishing and exploding gradients for deep models [4].
3. **He:** He initialization properly scales the backward signal by sampling initial weights from a mean-zero Gaussian distribution that has a variance of $\sqrt{2/n_L}$, where n_L denotes the size of the layer. This method allows deeper models to convergence [5].

3.4 Optimizer

Different optimizers may also influence the minimum that is found. We have tried using these different optimizers:

1. **Adam:** Adam is an algorithm for the first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. It is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters. Furthermore, the hyper-parameters have intuitive interpretations and typically require little tuning [2].

2. **RMSprop:** RMSprop is a optimization method that normalizes the current gradient by dividing by the moving average of squared gradients. This decreases the step for large gradients to avoid exploding, and increases the step for small gradient to avoid vanishing [3].

3.5 Loss function

This is a multi-class classification problem with 3 classes: Solid Normal Tissue, Lung Squamous Cell Carcinoma (LUSC), and Lung Adenocarcinoma (LUAD). Thus, a naturally good loss function is the cross entropy loss:

$$L_{x,y} = -\log\left(\frac{\exp(x[y])}{\sum_j \exp(x[j])}\right)$$

where x is an array that contains the probability predictions for all classes ($\sum_j x[j] = 1$), and y is the correct class.

Because we would like our model to output probability estimates, we used LogSoftMax as the final layer in the model with the negative log-likelihood loss. This is equivalent to the cross entropy loss.

3.6 Evaluation

3.6.1 Metric

Although cross-entropy loss is a good loss function for multi-class classification, the high level metric that we aim to maximize is AUC.

3.6.2 Method

While the neural network was trained on a per-tile basis, application of the model would be on a per-image basis. Thus, the scores from the neural network per tile must be aggregated for the final prediction. There are a few ways to implement this:

1. **Highest average score:** One can take the average score across all the tiles for each class, and predict the class with the highest average score.
2. **Most commonly predicted:** One can take the class with the highest probability for each tile to be the prediction for the tile. Then the aggregate prediction would be the class with the most tiles predicted for that class.
3. **Convolutional neural network:** Because the tiled scores can be seen as another small image, one can actually train another convolution neural network to perform the aggregate prediction.

4 Results

4.1 Completed tasks

4.1.1 Data processing

1. Uploaded all the images to NYU's HPC Prince, and tiled the images for lung cancer.
2. Created train, validation, and test data splits
3. Created a `dataloader` function that will output a tiled image with its corresponding label in batches.
4. Implemented the option to utilize the metadata as concatenated image channels.
5. Implemented the option to train with or without the data augmentation described in section 3.2.

4.1.2 Neural network training

1. Implemented options for training with Gaussian, Xavier, and He initialization.
2. Implemented the option to train with Adam or RMSProp as optimizers.
3. Implemented a few CNN architectures that successfully process our images and output the log probabilities for each class.
4. Implemented an early stop function that will stop training if no significant progress has been made on the validation loss within a certain number of checkpoints. Because evaluation takes a long time with the size of our dataset, we only evaluate with 20% of the validation set size after every 200 optimizer steps.

4.1.3 Model evaluation

1. Created an aggregation function based on what we described in section 3.6.2, except only with the **highest average score** and **most commonly predicted** methods.

We have run several experiments so far with the above implementations, including a 3 and a 5 layer Convolutional Neural Network using an Adam optimizer or He initialization as described above. Because our advisors want us to prioritize replicating their work, we have not yet experimented with the concatenated metadata or data augmentation (though the functionality has been tested and works).

5 Discussion

5.1 Problems to address moving forward

One of the issues we're currently facing is that although we have built a function for aggregating the tile probabilities into a prediction, this process takes a long time and we have not yet finished debugging it. This has prevented us from using this as an early stopping criterion, so we are using validation loss instead.

On a related issue, we still haven't been able to figure out how to compute AUC with multiple classes. With a one vs all approach, a probability distribution $[p_1\ p_2\ p_3]$ is [0.4 0.5 0.1] vs [0.4 0.3 0.3], would assign the same score of 0.4 to class 1 in both cases. We will be in contact with our advisors to ask them how they calculated AUC, so that we can compare our results with theirs.

Related to the other types of cancer (Breast and Kidney), we have just finished downloading and uploading all the data to NYU's Prince Cluster (more than 1.2 TB of data). We still need to tile these images and start testing different network implementations.

Finally, during our latest meeting with our advisors, they have recommended us to replicate the Inception V3 architecture using pretrained weights for the first layers and only training the last layer. We have found a Pytorch implementation online [6], but we still need to incorporate it with our training script and figure out how to do transfer learning.

References

[1] NCI's Genomic Data Commons https://gdc.cancer.gov/

[2] Kingma, D. and Ba, J. Adam: A Method for Stochastic Optimization https://arxiv.org/abs/1412.6980

[3] RMSprop

http://climin.readthedocs.io/en/latest/rmsprop.html

[4] Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf

[5] He, K. et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification https://arxiv.org/abs/1502.01852

[6] The torchvision package https://github.com/pytorch/vision/blob/master/torchvision/models/inception.py