

NAIST-IS-DD1561204

Doctoral Dissertation

**Implementation of Convolutional Networks on
Biomedical Images and its Connection to
Genomic Features**

Antonio Victor Andrew Asuncion

September 14, 2018

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of SCIENCE

Antonio Victor Andrew Asuncion

Thesis Committee:

Professor Shigehiko Kanaya	(Supervisor)
Associate Professor Naoaki Ono	(Co-supervisor)
Professor Yoshinobu Sato	(Co-supervisor)
Professor Md. Altaf-Ul-Amin	(Co-supervisor)

Implementation of Convolutional Networks on Biomedical Images and its Connection to Genomic Features*

Antonio Victor Andrew Asuncion

Abstract

A number of advances involving convolutional neural network (CNN) architectures have demonstrated that extracting morphological features from histological images can be effective for classification of subtypes of various diseases, especially cancer. On the other hand, varying types of gene expression or mutation data have been a rich and ubiquitous resource for studies involving cancer prognosis, survival, and others. In this study, we present a method for classifying transcriptome subtypes of lung adenocarcinoma from slices of pathological images whose features come from convolutional autoencoders pretrained on smaller images.

We also attempted to provide a stepping stone for whole slide image analysis by performing classification and correlation analysis. Whole slide images were processed based on features extracted from Google's Inception architecture. A classification was then implemented on the processed images. Furthermore, a correlation between the image features and their corresponding gene expression data were investigated.

Variants of autoencoders as building blocks of pretrained convolutional layers of neural networks were implemented on histological images gathered from the Cancer Genome Atlas database. From here, a sparse deep autoencoder was proposed and applied to images of size 2048x2048. We applied this model for feature extraction from pathological images of lung adenocarcinoma, which is comprised

*Doctoral Dissertation, Graduate School of Information Science,
Nara Institute of Science and Technology, NAIST-IS-DD1561204, September 14, 2018.

of three transcriptome subtypes. The sparse autoencoder network provided a 98.9% classification accuracy.

The second part of this study used features extracted from smaller images as a starting point for whole slide image analysis. We use 512x512 tiles as input for Google's Inception architecture, whose output is a 2048-dimensional vector for each tile. We applied dimension reduction to these vectors and correlated them with RNA-Seq data. A linear model was then created to further associate the features extracted from the aforementioned fully-connected layer to the gene expression values and use it for inference and other analysis.

The results showed that the larger input image that covers a certain area of the tissue is required to recognize transcriptome subtypes. This study exhibits the potential of autoencoders as a feature extraction paradigm and paves the way for a whole slide image analysis tool to predict molecular subtypes of tumors from pathological features. The analysis following the results of this model serves as a good starting point for outcome and prognostic predictions.

Keywords:

computer-aided diagnosis, convolutional neural networks, autoencoder, lung adenocarcinoma, gene expression

Contents

List of Figures	iii
List of Tables	1
1 Introduction	2
1.1 Background of the Study	2
1.2 Statement of the Problem	3
1.3 Significance of the Study	4
2 Review of Related Literature	6
2.1 Supervised Learning	6
2.1.1 Gradient Descent Approaches	6
2.2 Neural Networks	9
2.3 Convolutional Neural Networks	10
2.4 Autoencoders	12
2.5 Reconstruction Independent Subspace Analysis	13
2.5.1 Premise	14
2.5.2 Approach	14
2.6 Image Classification (Softmax Regression)	16
2.6.1 Softmax regression	16
2.6.2 MNIST Data	17
2.7 Classification of Lung Adenocarcinoma Transcriptome Subtypes	17
2.8 Medical Image Analysis	19
2.8.1 Medical Imaging and Computer-aided Diagnosis	19
2.8.2 Image Processing via CNNs	20
3 Materials and Methods	22
3.1 Classifier Variants	22

3.2	Towards Classification of Larger Images	23
3.3	Whole Slide Image Analysis	24
3.3.1	Transition from tiles to whole slides	25
3.3.2	Error Computation per sample	26
3.4	Interpretation of Features from CNN	26
3.5	Connecting Image Features to Genomic Features	27
4	Results and Discussion	31
4.1	Visualization of Filters	33
4.2	Internetwork Comparison	35
4.3	Deeper Networks	38
4.4	Whole Slide Image Analysis	40
4.4.1	Classification of Whole Slides	40
4.4.2	Interpretation of Classified and Misclassified Slides	43
4.4.3	Connecting Image and Genomic Features	44
5	Summary, Conclusions, and Recommendations	49
	References	52
	Publication List	57

List of Figures

1.1	Stages of Human Lung Adenocarcinoma [1]	3
2.1	Autoencoder model based on a convolutional neural network . . .	12
2.2	Sample of MNIST dataset [2]	17
2.3	Pathological images of lung adenocarcinoma subtypes and normal	19
2.4	Inception-v3 computational graph [3]	21
3.1	Pipelines for classifier variants	23
3.2	Structure of the whole network	24
3.3	Processing of Whole Slide Images	25
3.4	Image heatmap of whole slide	27
3.5	(Top) original histogram for RNA-Seq Data (Bottom) Histogram for remaining transcripts	28
3.6	Linear model pipeline	28
4.1	Example of the output of the autoencoder	33
4.2	Left: input image, Right: output of some encoding layers in the second autoencoder. The gradient from red to blue represents increase in signal intensity.	34
4.3	Examples of optimized local image for encoded outputs	34
4.4	Comparison of AE and RISA training.	35
4.5	Comparison Between Filters and Output of Networks	36
4.6	Training Accuracy for whole slide network	40
4.7	Error boxplots with Inception-v3 probabilities (left) and predicted probabilities (right)	41
4.8	ROC Comparison	42
4.9	Whole Slide and Corresponding Probability Visualization	44

4.10 Gene expression boxplot with predicted expression overlay	45
4.11 <i>t</i> -SNE plot for output of fully-connected layer	46
4.12 <i>t</i> -SNE plot with clusters and samples	46
4.14 <i>t</i> -SNE boxplot for general dataset and 15 driver genes	48

List of Tables

3.1	Sample error computation	26
4.1	Confusion Matrices for Varying Networks	32
4.2	Confusion Matrix for RISA	33
4.3	Subtype classification accuracy tables for varying networks and filter sizes	37
4.4	Confusion matrices and accuracy for 128px, 512px, and 2048px experiments	39
4.5	Testing Accuracy for whole slide network	41
4.6	Slide Information	43

1 Introduction

1.1 Background of the Study

Machine learning continues to be a vital innovation used in several fields. Rapid development of machine learning algorithms brings us a wide range of applications for image recognition and classification. In particular, a significant advancement of visual recognition using deep learning architectures has been shown by the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [4], which has served as a testbed for a few generations of large-scale image classification systems. A convolutional neural network (CNN) provides a promising architecture that can extract features from given images automatically, optimize the manifold of image space, and show great success in image classification and medical image analysis [5, 6].

From a biology standpoint, these methods can be used for gene expression interpolation and classification of several data sets. In recent years, a number of advances involving CNN architectures have been developed and implemented for computer-aided diagnosis [3]. These studies have demonstrated that extracting morphological features from histological images can be effective for classification of subtypes of various diseases, especially cancer. On the other hand, varying types of gene expression or mutation data have been a rich and ubiquitous resource for studies involving cancer prognosis, survival, and others. Furthermore, it has been an important instrument for image classification and inference as of late. Image classification and analysis has been an important achievement of computational systems, and in fact, it is still a growing, revolutionary field. Specifically, being able to perform analysis on pathological images proves to be vital for medicine and bioinformatics. Any histological image patch is rich in content because of several components (cell type, organization, state and health,

secretion). This can be analyzed manually by pathologists and biologists. (See Fig. 1.1)

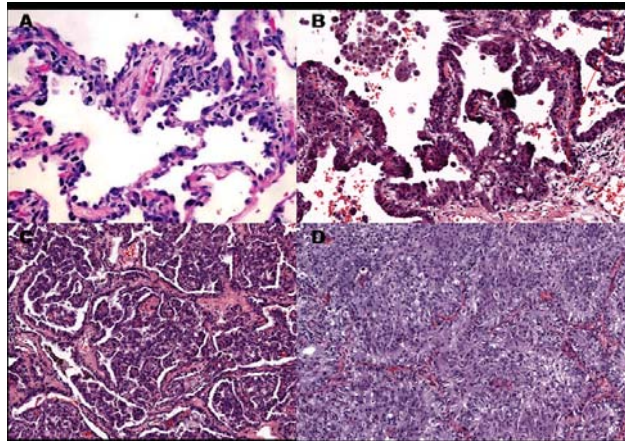


Figure 1.1: Stages of Human Lung Adenocarcinoma [1]

On the other hand, the potential of gene expression profiles as a source of relevant information has already been well documented rich resource for predicting outcome and treatment [7, 8]. This set of details, combined with other types of data, like mutation, even allows for an even richer source of information [9]. With the evolution of methods that extract image features, we now have the resources for possible connecting information learned from images and information learned from gene expression data.

1.2 Statement of the Problem

Image processing methods using deep neural networks are emerging very rapidly and these architectures were developed for general image analysis such as photo classification, face recognition, among others. However, analysis of biomedical images requires a more specific viewpoint focusing extensively on their biological features. So the first aim of this work is to analyze current algorithms for the classification of biomedical images by extracting useful biological features using the concept of deep neural networks, one of the classic tools used in machine learning. Some of the aforementioned features include cell patterns that do not always appear on general image classification but occurs commonly in the image

of target organisms. Those features must be extracted effectively from the given training data in order to classify and analyze the images efficiently. In this study, we propose an application of CNNs for feature extraction and classification of lung adenocarcinoma pathological images, and use the learned features for classification of large image data. Information gathered from these features were used for the implementation of several experiments involving whole slide image analysis. These analyses involved not only classification but also an initial approach to correlation with gene expression profiles. Specifically, our analysis involves classification of whole slides based on tumor/normal labels, molecular subtypes, among others. Moreover, we also want to identify how classification is arrived at. The method proposed here took into consideration the features extracted from several convolutional neural network architectures to determine some basis for comparison.

Explicitly, the formulated algorithms were implemented on lung cancer pathological images. These approaches use small images as input, usually less than 300px by 300 px. However, whole slide images gathered by The Cancer Genome Atlas (TCGA) network are of a much larger magnitude. This study presents an approach that transfers information learned from small input images to larger input data. By applying unsupervised learning through autoencoders, we will be able to extract features that are not heavily reliant on classification information.

Furthermore, we would like to explore the correlation between features extracted from whole slide images and their corresponding gene expression features. We would like to determine if features extracted from images include information about gene expression patterns, mutations, among others. So another aim of this study is to establish a linear model to determine the correlation between the features extracted from histological images through CNNs and their corresponding gene expression data.

1.3 Significance of the Study

We want to understand the internal mechanisms of several variations of convolutional neural network-based architectures. Specifically, we want to determine the properties or characteristics emphasized by these networks by looking into the

features extracted from the convolutional filters. In doing so, we may be able to understand the underlying properties of biomedical images that will serve some purpose in distinguishing these images from one another.

Whole slides are approximately 40000x40000 pixels large. Difficulties may arise from analyzing a huge amount of data. Our method can serve as a blueprint for whole slide image analysis, which gives a more comprehensive view of the image. The final goal here is to initiate an approach that connects image features to genomic features like RNA-Seq, mutation, among others. While genomic information have been studied widely, studying histological images are still relatively new. This study contributes to this currently evolving field by first implementing CNN architectures involving whole slide images and by using information obtained from those architectures for an initial correlation analysis with genomic data.

2 Review of Related Literature

2.1 Supervised Learning

Suppose we are given m training examples $(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})$, where each $\vec{x}^{(i)}$ is an n -dimensional vector representing n features for the training data, and $y^{(i)}$ is the corresponding output for each sample $\vec{x}^{(i)}$. Our goal is to determine the correlation between the training input and output for either inference or classification. Specifically, we will represent this correlation through a function h . For this section, we choose a linear function for h such that our goal is to find $\vec{\theta} = \langle \theta_0, \theta_1, \dots, \theta_n \rangle$ such that for an input vector $\vec{x} = \langle 1, x_1, x_2, \dots, x_n \rangle$, we have

$$h_{\vec{\theta}}(\vec{x}) = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n, \quad (2.1)$$

and $h_{\vec{\theta}}$ minimizes the standard error function defined by

$$C(\vec{\theta}) = \frac{1}{2} \sum_{i=1}^m (h_{\vec{\theta}}(x^{(i)}) - y^{(i)})^2. \quad (2.2)$$

Remark 1. If we define $x_0 = 1$, then we can even say that $h_{\vec{\theta}}(\vec{x}) = \vec{\theta}^T \vec{x}$, where $\vec{\theta}^T$ is the matrix transpose of the vector $\vec{\theta}$.

Remark 2. Note that Equation Eq. (2.2) is a convex quadratic function, which means that it will have a local extrema, which consequently is a global extrema. Specifically, this extreme value will be a minimum.

2.1.1 Gradient Descent Approaches

Batch Gradient Descent

The general method that we will look at is called the **batch gradient descent** algorithm.

We will determine the values of each of the θ_j 's individually using the method of steepest descent, or a gradient descent algorithm, whose goal is to minimize $C(\vec{\theta})$.

Numerically, this means that we need to produce iterations of $\vec{\theta}$ such that we can decrease $\frac{\partial}{\partial \theta_j} C(\vec{\theta})$.

For the algorithm, we will make use of the notion of the learning rate.

Definition 3. The **learning rate**, usually denoted by α , is a parameter which determines how much recent information will affect old information. If $\alpha = 0$, then new information will not affect old information. On the other hand, if $\alpha = 1$, then only the most recent information will be considered.

So if $\theta_j^{(k)}$ is the current iterate for the j th parameter, then

$$\theta_j^{(k+1)} := \theta_j^{(k)} - \alpha \frac{\partial}{\partial \theta_j} C(\vec{\theta}).$$

Here, α is the value of the predetermined learning rate. By convention, we use $\alpha = 0.1$.

If we follow the data as in Eq. (2.1) and Eq. (2.2), then for each j ,

$$\begin{aligned} \frac{\partial}{\partial \theta_j} C(\vec{\theta}) &= \frac{\partial}{\partial \theta_j} \left[\frac{1}{2} \sum_{i=1}^m (h_{\vec{\theta}}(x^{(i)}) - y^{(i)})^2 \right] \\ &= \sum_{i=1}^m \frac{1}{2} \cdot 2 (h_{\vec{\theta}}(x^{(i)}) - y^{(i)}) \frac{\partial}{\partial \theta_j} (\theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_j x_j^{(i)} + \cdots + \theta_n x_n^{(i)} - y^{(i)}) \\ &= \sum_{i=1}^m (h_{\vec{\theta}}(x^{(i)}) - y^{(i)}) x_j^{(i)}. \end{aligned}$$

This means that the update algorithm for θ_j , $0 \leq j \leq n$, is given by

$$\theta_j^{(k+1)} := \theta_j^{(k)} - \alpha \sum_{i=1}^m (h_{\vec{\theta}}(x^{(i)}) - y^{(i)}) x_j^{(i)}. \quad (2.3)$$

Numerically, we perform the following steps.

1. Provide a required tolerance level $\epsilon > 0$, and a maximum number of iterations.
2. If $C(\vec{\theta}) \leq \epsilon$, return $\vec{\theta}$.

3. Otherwise, for each j , perform Eq. (2.3) until the maximum number of iterations is exceeded.

We can perform this algorithm until it converges, that is, if it reaches a value less than some predetermined tolerance level, or until it achieves a preset number of iterations, at which point, we say that $\theta_j^{(k)}$ diverges.

Stochastic Gradient Descent

Although the batch gradient descent algorithm generally converges to a minimum, the computational cost is rather expensive because we have to go through all the training data before we can update each of the parameters θ_j . An improvement of this method is called the **stochastic gradient descent** algorithm, wherein we perform the update algorithm after an occurrence of individual data. This means that our update algorithm, for a specific data point $(x_j^{(i)}, y^{(i)})$, will just be

$$\theta_j^{(k+1)} := \theta_j^{(k)} - \alpha (h_{\vec{\theta}}(x^{(i)}) - y^{(i)}) x_j^{(i)}. \quad (2.4)$$

We are still following the batch gradient descent approach, but this time, we follow these steps.

1. Provide a required tolerance level $\epsilon > 0$, and a maximum number of iterations.
2. If $C(\vec{\theta}) \leq \epsilon$, return $\vec{\theta}$.
3. Otherwise, for each j :
 - a) for each m , perform Eq. (2.4)
 - b) If $C(\vec{\theta}) \leq \epsilon$, return $\vec{\theta}$.
 - c) Execute until loop is finished.
4. Execute until number of iterations is exceeded.

2.2 Neural Networks

Our goal here is to apply the concepts in the previous section and develop an algorithm that will work on neural networks. A neural network (NN) is a statistical model that simulates the movement of individual components, called neurons and are used to determine a well-defined function h that will best fit the data.

So, much like in the previous section, suppose we are given m training data points $(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})$. This time, we specify the output values $y^{(i)}$ to either be 0 or 1.

Suppose we have a model where each sample is represented by n features denoted by x_1, x_2, \dots, x_n . As in the previous section, we let $x_0 = 1$ be the intercept term. As a consequence, if we have a specific input vector $\vec{x} = \langle x_1, x_2, \dots, x_n \rangle$, then the input of our function can be $z = \vec{\theta}^T \vec{x} = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$.

Remark 4. As a convention that will be used here, for $a \in \mathbb{R}$ and $\vec{v} = \{v_1, v_2, \dots, v_n\} \in \mathbb{R}^n$, we have $\langle a, \vec{v} \rangle = \langle a, v_1, v_2, \dots, v_n \rangle$.

Let the function $h_{\vec{\theta}}(\vec{x})$ be the nonlinear form of the hypothesis that will be used for data-fitting, regression, and specifically, for our purposes, classification.

Conventionally, we can choose

$$f_1(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}},$$

whose range is $[-1, 1]$, or the sigmoid function defined by

$$f_2(x) = \frac{1}{1 + e^{-x}},$$

whose range is $[0, 1]$.

f_1 and f_2 are examples of **activation functions**.

For our purpose, we use $f(x) := f_2(x)$, since we want to have our training output to be either 0 or 1.

This means that we can have

$$\begin{aligned} h_{\vec{\theta}}(\vec{x}) &= f(\vec{\theta}^T \cdot \langle x_0, \vec{x} \rangle) \\ &= \frac{1}{1 + \exp(\vec{\theta}^T \cdot \langle x_0, \vec{x} \rangle)}. \end{aligned}$$

Initially, the input can proceed directly to activation function, which means we can just follow the algorithm as in the previous section. However, the main advantage of utilizing NN is that we can have multiple layers of activation. That is, instead of proceeding directly to the activation function, we can allow the input to go through other layers, called **hidden layers**, as part of the process. This converts the input into neurons that may be a more viable source of features for analysis.

Let n_l be defined to be the number of layers of the network. s_l be the number of neurons or nodes in layer l . Let $a_i^{(l)}$ be the output value of unit i in layer l . This means that, in the input layer, we have $a_i^{(1)} = x_i$, and in the output layer, we have $a^{(n_l)}$ as the final output value of the network.

Consequently, we now have $\Theta = \{\Theta^{(1)}, \Theta^{(2)}, \dots, \Theta^{(n_l-1)}\}$ as the set of parameters for the network, with each of the Θ_i the $s_{i+1} \times s_i$ matrix representing the connection between the input of layer i and the nodes of layer $i+1$. For example, $\Theta^{(1)}\vec{x}$ will be a s_2 -dimensional vector representing the input for that layer, except that we retain $x_0 = 1$ for this layer.

We still apply the cost function Eq. (2.2), but this time, incorporate a **regularization term** that will decrease the magnitude of the weights, which consequently prevents overfitting. We then have

$$C(\Theta) = \frac{1}{2} \sum_{i=1}^m \left(h_{\Theta}(x^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \left(\sum_{i=1}^{s_{l+1}} \sum_{j=1}^{s_l} \Theta_{ij}^{(l)} \right). \quad (2.5)$$

The goal is to determine Θ such that $C(\Theta)$ is a minimum.

We start with random values of $\Theta_{ij}^{(l)} \sim N(0, \epsilon^2)$. Conventionally, $\epsilon = 0.1$ is used.

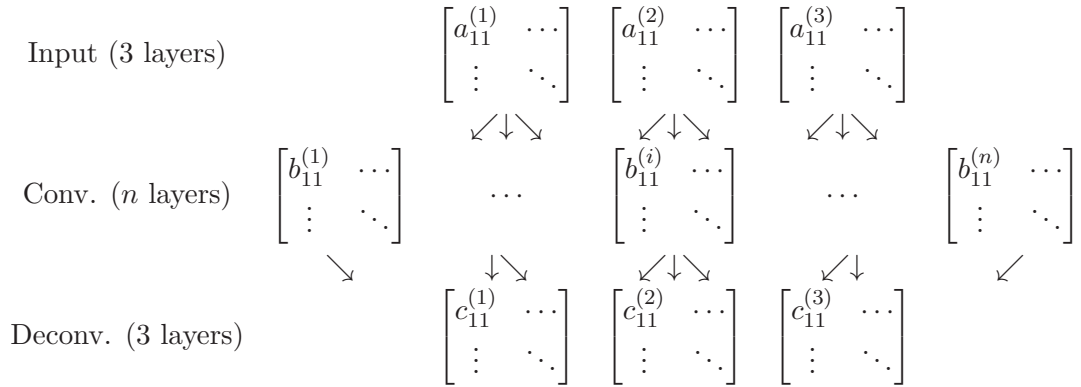
2.3 Convolutional Neural Networks

Definition 5. Convolution describes the use of matrices (convolutional filters) that can be passed through the image for feature extraction. For example, a matrix C can be passed through the input such that if \vec{x} is the n -dimensional input vector of the network, then the encoded images (H) is given by

$$H = C_{k \times n} \vec{x},$$

where k is the desired output dimension. To reverse the process, we apply H^T to the encoded images. This process is called **deconvolution**.

Example 6. Consider the following diagram:



Definition 7. Pooling: method of combining regions of neurons into a single layer, mainly used for size reduction, while keeping the important features.

- Max-pooling (standard)

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \implies [\max(a_{11}, a_{12}, a_{21}, a_{22})]$$

- Mean-pooling

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \implies \left[\frac{a_{11} + a_{12} + a_{21} + a_{22}}{4} \right]$$

To reverse the process, we perform unpooling.

$$\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \implies \begin{array}{|c|c|} \hline b_{11} & b_{11} \\ \hline b_{11} & b_{11} \\ \hline b_{21} & b_{21} \\ \hline b_{21} & b_{21} \\ \hline \end{array} \begin{array}{|c|c|} \hline b_{12} & b_{12} \\ \hline b_{12} & b_{12} \\ \hline b_{22} & b_{22} \\ \hline b_{22} & b_{22} \\ \hline \end{array}$$

Remark 8. In general, the process would follow this diagram:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \xrightarrow{\text{Pooling}} [a] \xrightarrow{\text{Unpooling}} \begin{bmatrix} a & a \\ a & a \end{bmatrix}$$

2.4 Autoencoders

An autoencoder is an unsupervised machine learning architecture that extracts characteristic features from given inputs by learning a network which reproduces input data from those features. Fig. 2.1 shows the basic design of the autoencoder used in our model. The input data is scanned by a convolutional filter and down-sampled by a max-pooling layer then passed on to an encoding layer. The output here can then be used to generate the input data using the reversed network. The total network is optimized to minimize the difference between input and output data.

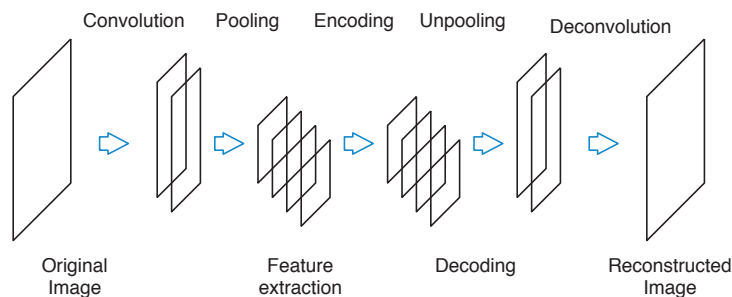


Figure 2.1: Autoencoder model based on a convolutional neural network

To enhance the efficiency of feature extraction and information compression in the autoencoder, we introduced a sparsity penalty. We compute information entropy of the output of the encoding layer and add the penalty for the optimization function (L) to minimize the effect of overfitting. The optimization function is defined as follows:

$$L = R + \lambda_s S, \quad (2.6)$$

where

$$R = \sum_i^N (x_i^{\text{output}} - x_i^{\text{input}})^2 \quad (2.7)$$

and

$$S = \sum k = 1^n \sum_{j=1}^M \left(-r_j^{\text{encode}} \log r_j^{\text{encode}} \right). \quad (2.8)$$

Here, r_j^{encode} is the output intensity of filter j in the encoding layer relative their total summation. N and M are the numbers of nodes in the input and encoding layers, respectively, and λ_s is a weight constant.

Stacked autoencoders allow us to extract more complex image features with higher-order structures, while some detail information will be lost in down-sampling. It is worth noting that stacked autoencoders can be trained independently. That is, the network of the first autoencoder can be fixed after training and left aside when we train the network for the second optimizer. This reduces the number of trainable parameters and required computation. Special types of autoencoders have shown to be a good foundation for feature extraction [10]. In this study, a method that incorporates pretraining of the autoencoder network similar to the method by Hinton and Salakhutdinov [11] was proposed. Specifically, their pre-training is composed of learning layers of restricted Boltzmann machines as a way to efficiently fine-tune a network for reconstruction and classification purposes.

2.5 Reconstruction Independent Subspace Analysis

Part of the goal of this work is to compare several variants of classifiers. One specific network that will be used here is based on a specific type of autoencoder. Le, et al. [12] presented a novel method for automated feature detection and extraction of unlabeled image patches using a technique called reconstruction independent subspace analysis (RISA). We will look into incorporating said method into our classification. Specifically, we will be looking at lung cancer images and the main goal of the study is to perform analysis on the features gathered from these images. There is a rising need for algorithms that can address the problems discussed earlier. And previous work have recurring themes with regard to this.

1. Tumor Grading: segmentation is the answer. Pathologists can assess the

situation, but it is rather difficult because of mixed grading. Many studies now use segmentation to address the problem.

2. Region-based analysis is another commonality. The main feature detection process involves defining color and texture features for different classification methods.

2.5.1 Premise

Other than those mentioned above, there is a need for an algorithm that quantifies big datasets quickly. In [12], the main concept is the use of reconstruction independent subspace analysis (RISA). One of its main features is translation invariance. That is, if the input data is translated by one or two pixels, the value of the linear filters change, but the value of the feature that averages their values will only change slowly. Some of the problems addressed are:

1. does it capture complex tumor signatures that are necessary for tumor grading?
2. does it remain stable through multiple sets of testing and training data?

2.5.2 Approach

The main notion used is called Reconstruction Independent Subspace Analysis (RISA).

Basically, RISA is an unsupervised learning algorithm. It is described as a two layer network (including input) wherein the responses are nonlinear. Specifically, we have square and square root nonlinear responses for the first and second layer, respectively.

Two sets of parameters, W and V , are used for feature extraction. W is learned in the conventional manner, while V is fixed and is used to represent the subspace structure of the neurons in the first layer.

What is done is each of the hidden units in the second layer pools over a small neighborhood of adjacent first layer units.

Let $\vec{x}^{(t)}$ be defined as the t th input (out of a group of T), each of which is an n -dimensional feature vector. Let there be k neurons in Layer 1, and m neurons in Layer 2.

As a consequence, we have $W \in \mathbb{R}^{k \times n}$, while $V \in \mathbb{R}^{m \times k}$.

$$\begin{aligned}
 W_{m \times k} &= \begin{bmatrix} \vec{w}_1^T \\ \vec{w}_2^T \\ \vdots \\ \vec{w}_k^T \end{bmatrix} \\
 &= \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & & \ddots & \vdots \\ w_{k1} & w_{k2} & \cdots & w_{kn} \end{bmatrix}.
 \end{aligned}$$

On the other hand,

$$V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1k} \\ v_{21} & v_{22} & \cdots & v_{2k} \\ \vdots & & \ddots & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{mk} \end{bmatrix}.$$

Recall that we want our responses to be nonlinear. Specifically, in the first layer, we want a square response, while in the second layer, we have a square root response. This means that in Layer 1, if we define the response to the sample $\vec{x}^{(t)}$ to be given by the k -dimensional vector \vec{p} , we then have

$$\begin{aligned}
 \vec{p}(\vec{x}^{(t)}; W) &= W\vec{x}^{(t)} \circ W\vec{x}^{(t)} \text{ (element-wise multiplication)} \\
 &= \begin{bmatrix} \left(\sum_{l=1}^n w_{1l}\vec{x}_l^{(t)} \right)^2 \\ \left(\sum_{l=1}^n w_{2l}\vec{x}_l^{(t)} \right)^2 \\ \vdots \\ \left(\sum_{l=1}^n w_{kl}\vec{x}_l^{(t)} \right)^2 \end{bmatrix}
 \end{aligned}$$

That means if we want to extract the j th term, we have

$$\vec{p}_j(x^{(t)}; W) = \left(\sum_{l=1}^n w_{jl} x_l^{(t)} \right)^2.$$

Now, for the second layer, we want a square root response represented by an m -dimensional vector, which we denote by \vec{q} . This means that

$$\begin{aligned} \vec{q}(x^{(t)}; W, V) &= V \vec{p}(x^{(t)}; W)^{.1/2} \text{ (element-wise square root)} \\ &= \begin{bmatrix} \sqrt{\sum_{r=1}^k v_{1r} \vec{p}_r(x^{(t)}; W)} \\ \sqrt{\sum_{r=1}^k v_{2r} \vec{p}_r(x^{(t)}; W)} \\ \vdots \\ \sqrt{\sum_{r=1}^k v_{mr} \vec{p}_r(x^{(t)}; W)} \end{bmatrix} \end{aligned}$$

2.6 Image Classification (Softmax Regression)

2.6.1 Softmax regression

The main advantage of using softmax regression is that we can now determine the probability of our image being in each of the different classes, rather than simply classifying it in a single class.

Definition 9. Cross-entropy:

$$H_{y'}(y) = - \sum_i y'_i \log(y_i)$$

where y is the predicted probability distribution, y' is the true distribution.

This is a scheme used to evaluate the accuracy of the prediction scheme.

We apply a linear operator $W_{m \times n}$ on an input vector \vec{x} , where n is the dimension of \vec{x} , and m is the dimension of the one-hot vector y corresponding to the input.

An m -dimensional vector \vec{b} will also be added similar to the standard NN approach.

Then the actual input in the regression will then be $\vec{e} = W\vec{x} + \vec{b}$.

The actual softmax function will then be

$$\vec{y} = \text{softmax}(Wx + b)$$

2.6.2 MNIST Data

MNIST is an image dataset consisting of handwritten digits.



Figure 2.2: Sample of MNIST dataset [2]

The MNIST dataset has been used as one of the standard input for benchmarking of new networks, which was initiated by LeCun et al. [2].

This study relies on an investigation conducted by Masci, et al. to determine the ability of standard CNNs to extract features using an unsupervised learning method [13]. They used the MNIST dataset, which has been the standard for this type of testing. We see in this work the ability of pre-training a network for reconstruction to still be capable of performing classification. They were able to extract features from unlabeled data, which when combined with backpropagation algorithms can still become efficient classifiers.

2.7 Classification of Lung Adenocarcinoma Transcriptome Subtypes

Lung cancer is the leading cause of cancer-related mortality, and adenocarcinoma is its most common histological subtype [14, 15]. The overall prognosis for lung cancer remains poor, despite recent advances in molecular targeted therapies. Several cancer genome projects have analyzed cohorts of lung cancer patients and revealed genome and transcriptome alterations. Most recently, the Cancer Genome Atlas (TCGA) has described the comprehensive genomic landscape of

lung adenocarcinoma in a large cohort [16]. These studies not only elucidated oncogenic mechanisms but also shed light on previously unappreciated heterogeneity of gene expression profiles. As a consequence of genomic alterations and gene mutations in cancer cells, aberrant patterns of gene expression profiles occur, which eventually determine cancer cell behaviors. In line with this, it is worth noting that the aforementioned TCGA study has identified three transcriptome subtypes of lung adenocarcinoma: the terminal respiratory unit (TRU, formerly bronchioid), the proximal-proliferative (PP, formerly magnoid), and the proximal-inflammatory (PI, formerly squamoid) transcriptional subtypes [17]. It has been further demonstrated that this classification is associated with clinical features and gene mutation profiles. In terms of morphological features, lung adenocarcinomas display high inter-individual and intra-tumoral heterogeneity. However, it remains undetermined whether the transcriptome subtypes are associated with distinctive patterns of pathological findings. If it is the case, image analyses on biopsy of resected tissue samples will be helpful to infer transcriptional changes in tumor tissues, which can assist precise diagnosis and clinical decision making. In this study, we propose a model to classify three lung adenocarcinoma transcriptome subtypes from their pathological images using a deep learning approach. Fig. 2.3 shows a sample of each of the three subtypes alongside four different samples of normal images.

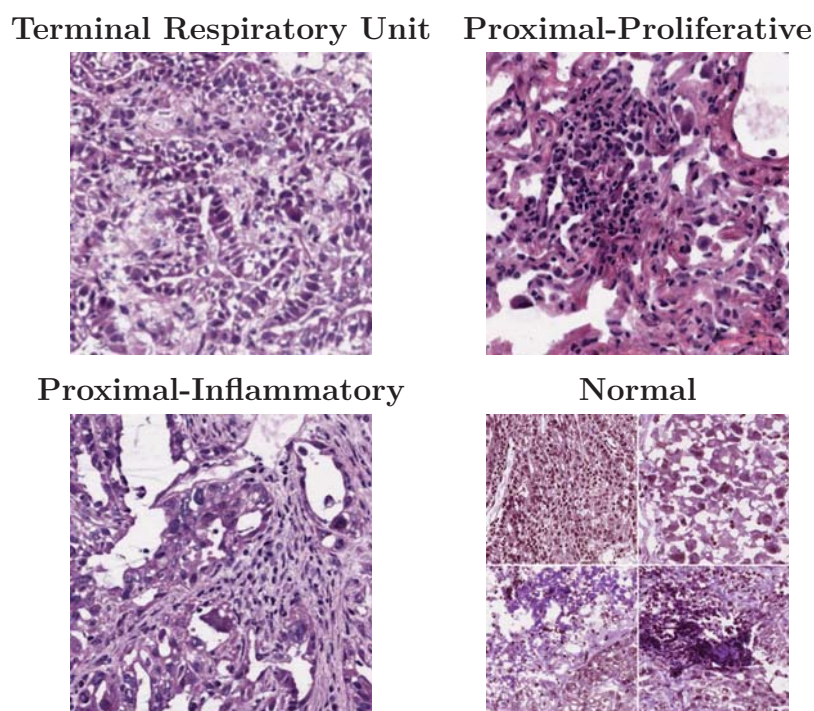


Figure 2.3: Pathological images of lung adenocarcinoma subtypes and normal

2.8 Medical Image Analysis

2.8.1 Medical Imaging and Computer-aided Diagnosis

Medical image interpretation is a crucial part of the diagnostic process. Current advances in artificial intelligence [18] can now automate the process of providing assistance to radiologists and pathologists in this light.

By combining these innovations with imaging techniques like X-ray and MRI, computer-aided diagnosis (CAD) proves to be a vital contributor in the field of cancer detection, disease assessment, among others [19].

An article by Doi [20] provides a detailed history of medical imaging in CAD. It mentions how the implementation of CAD schemes that use medical images improves the assessment performance in several detection problems such as lung nodules and vertebral fractures. Moreover, studies that involve automatic pattern recognition or classification from various types of images [21,22] contribute to an

effective and more efficient disease evaluation and analysis.

2.8.2 Image Processing via CNNs

Machine learning continues to be a vital innovation used in several fields. From a biology standpoint, it can be used for gene expression interpolation and classification of several data sets. Moreover, it has been an important instrument for image classification and inference in recent years. On the other hand, image classification and analysis has been an important achievement of computational systems in recent times, and in fact, it is still a growing, revolutionary field. Specifically, being able to perform analysis on pathological images proves to be vital for medicine and bioinformatics. Image processing methods using deep neural networks are currently developing very rapidly. However, those approaches mainly target general image analysis such as photo classification, face recognition, among others. An analysis of biomedical images requires a more specific viewpoint focusing extensively on their biological features [23–26].

We see from [12] a specific type of CNN using the notion of a reconstruction independent subspace analysis (RISA). This is an unsupervised learning method for the reconstruction of images that emphasizes the invariance between the extracted features, which means that neighboring filters are designed to share the same property. They were able to show that these invariant features are vital in the classification of images if we attach a supervised layer to the pretrained RISA network.

Since part of our goal is an understanding of the internal architectures of several CNN variations, we also look at [27]. Their work provides a number of visualization experiments for this purpose, and we can follow a similar approach for our data.

We also constructed a deep learning model of the sparse autoencoder (SAE) for the differentiation of the distinct types of lung adenocarcinoma from pathological images [28, 29].

Furthermore, machine learning and computer vision has provided us powerful methods to improve accuracy and efficiency of image classification. These methods rely upon manually curated image features to characterize specific features of tumors. However, recent development of approaches like deep neural networks

allows us to extract image features from given data automatically, without using hand-made features. Using pretrained neural networks, we can extract features of tumors and distinguish them according to their shapes. However, when we address the classification of adenocarcinoma subtypes, local features of cell shapes are not enough to describe the variation and distribution of various cells in the tissue [30]. In this paper, we propose variations of CNNs that uses multiple reduction layers in order to evaluate a large area of pathological images and classify lung adenocarcinoma subtypes.

Finally, one of the major breakthroughs in image processing is the Inception [3] architecture, whose computational graph is shown below:

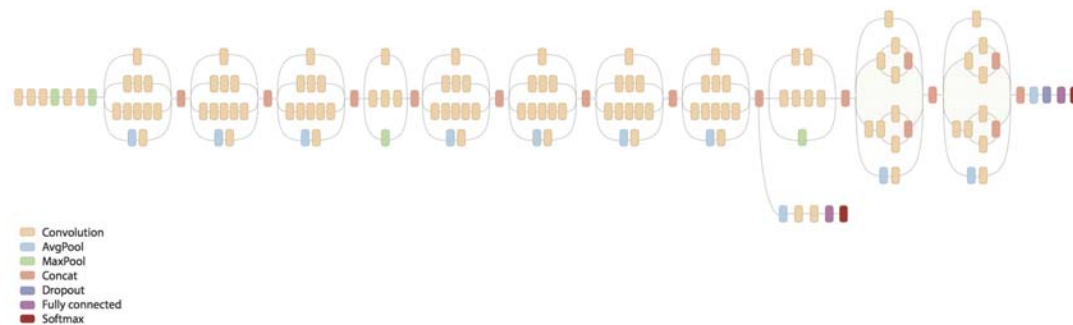


Figure 2.4: Inception-v3 computational graph [3]

Fig. 2.4 shows an intricate image processing network, composed of several stacks of standard convolution and pooling, interspersed between concatenation and partition layers. It is composed of 48 layers and around 25 million parameters. This will be used as a basis for feature extraction in this study.

3 Materials and Methods

3.1 Classifier Variants

In the first part of this study, we implemented three types of classifiers and compared their corresponding results. These networks can be distinguished based on how the convolutional filters were learned and extracted.

We call the first network a direct classifier and it is described by a convolutional network attached to a softmax classification layer. The features were extracted according to optimal classification. Softmax cross-entropy was used as the loss function.

On the other hand, the subsequent networks are pretrained autoencoder CNNs. The final layers of these networks were attached to a softmax classification layer, and the features for these specific part of the network were extracted similar to the direct classifier.

In particular, the second network is a pretrained autoencoder whose features are extracted following the reconstruction paradigm R from Eq. (2.7).

On the other hand, the third network is a pretrained reconstruction independent subspace analysis (RISA) network. It is a two-layer autoencoder variant composed of convolution and pooling layers. The main distinction of a RISA network is that it emphasizes minimal translational invariance [12]. If we denote the learned matrix from the convolutional layer as C , and the fixed matrix for the pooling layer as H , then for an input vector \vec{x} , the second layer output is

$$p_i(\vec{x}; C, H) = \sqrt{\sum_{m=1}^k H_{im} \left(\sum_{j=1}^n C_{mj} \vec{x}_j \right)^2}.$$

The features extracted from a RISA network will be learned through the following

heuristic:

$$\arg \min_C \sum_{t=1}^T \left(\frac{1}{T} \|CC^T \vec{x}^{(t)} - \vec{x}^{(t)}\|^2 + \lambda \sum_{i=1}^k p_i(\vec{x}^{(t)}; C, H) \right),$$

where $\{\vec{x}^{(t)}\}_{t=1}^T$ is the input dataset, and λ a weight constant.

This rule extracts features less expensively than manually-designed feature extraction methods.

Fig. 3.1 shows an overview of the different pipelines for the three variants. Here, the softmax classifier takes logistic outputs.

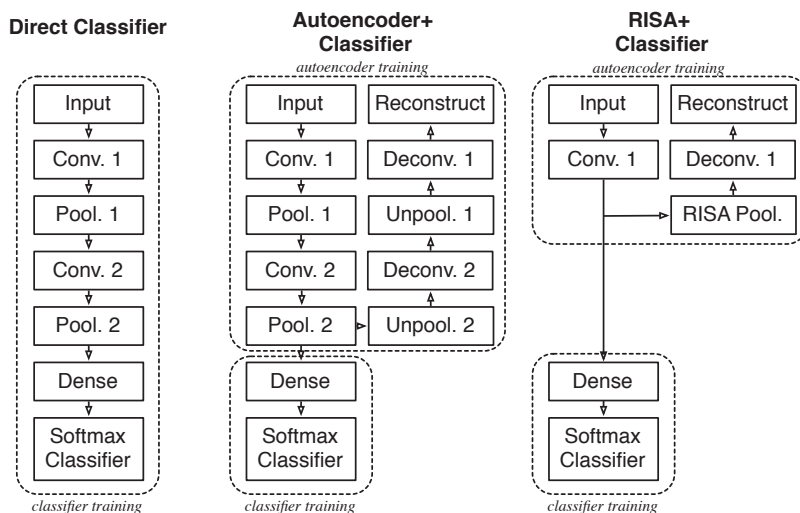


Figure 3.1: Pipelines for classifier variants

3.2 Towards Classification of Larger Images

In the second part of this study, we constructed a model based on three autoencoders and one classification reducer that takes logistic outputs. Fig. 3.2 shows the structure of the network. 2048px \times 2048px slices from the pathological images were used as input for the first autoencoder. For an initial feature extraction, we first pretrain three stages of convolutional autoencoder. The output from the encoding layer of the third autoencoder is passed to the reduction classifier. Since the size of the third encoding layer is still large, we divided it into 16 \times 16 subpanes, and in each subpane, the input from the encoding layer is reduced to 24

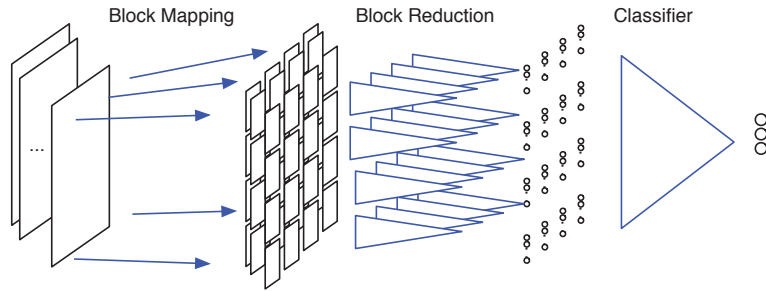


Figure 3.2: Structure of the whole network

output nodes through fully connected layers. Note that all the subpanes share the same reduction network, in other words, it is also a convolution without overlap between windows. Finally, the output of the reduction layer is reduced again into three nodes which represent the three classes of lung adenocarcinoma subtypes. Using multiple reduction layers, we can evaluate larger pathological images in order to recognize the features based from cell distribution in the cancer tumor and classify their transcriptome subtypes. The network in this model is composed of 11 layers and 97227 nodes in total. We implemented these networks based on python using TensorFlow [31] libraries, which provides various basic functions for neural networks and machine learning algorithms. This time we incorporate the sparsity penalty as described in Equation Eq. (2.8) to extract features.

The actual dataset is composed of pathological images of lung adenocarcinoma from The Cancer Genome Atlas (TCGA) [32, 33]. There are 409 whole slides from 230 cancer patients which are classified into three transcriptome subtypes according to their gene expression patterns. The original pathological slide images have a resolution of 20000–40000 pixels, whose actual sizes are approximately 1–2 cm². We randomly clipped the original images into 2048px×2048px slices and obtained 106505 slices as the input data for our models.

3.3 Whole Slide Image Analysis

For this section, we now change the dataset to prostate adenocarcinoma whole slide images, still extracted from the TCGA collection [33].

3.3.1 Transition from tiles to whole slides

As input, we use 704 whole slide images from TCGA-PRAD, 60% of which goes to training. Because of the small number of input data, We also implement some data augmentation as part of the preprocessing (like rotations and flips), which results to having four times the original amount of input.

Fig. 3.3 describes an example of the process that the whole slides will go through for analysis. Fig. 3.3a is a sample whole slide, while Fig. 3.3b shows the standard division into tiles. Finally, Fig. 3.3c is a visualization of the probability matrix emanating from the tumor probability of each tile.

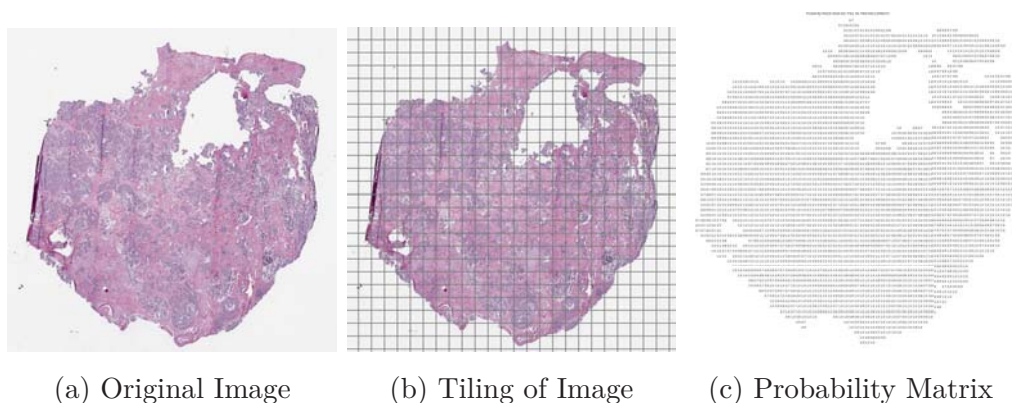
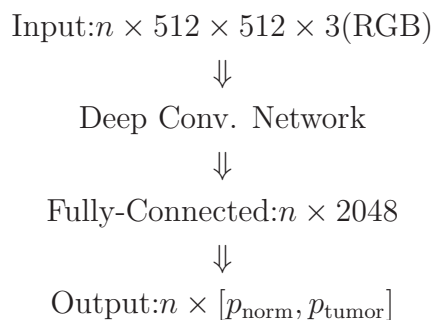


Figure 3.3: Processing of Whole Slide Images

We then use a convolutional network attached to a softmax classifier that will label each whole slide with its corresponding annotation.

The architecture of the whole slide network, consisting of 4 stages of convolution, followed by fully-connected and softmax classification layers, is shown below:



3.3.2 Error Computation per sample

We then compute several values for each sample as seen in Table 3.1. The first three columns indicate the patient ID, actual tumor percentage, and actual annotation gathered from the database, respectively.

The “Mean p ” column indicates the mean tumor probability of the tiles for each whole slide.

The last two columns are computed from the convolutional classifier.

The “Pred. Label” column indicates the label predicted by the convolutional classifier for each whole slide.

Finally, the “Pred. p ” column indicates the probability of the whole slide being a tumor image, as computed by the classifier.

Table 3.1: Sample error computation

Patient	Tumor %	Actual Label	Mean p	Pred. Label	Pred. p
1	50	1 (Tumor)	0.4	1	0.8
2	90	1 (Tumor)	0.95	1	0.99
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
n	0	0 (Normal)	0.1	0	0.005

3.4 Interpretation of Features from CNN

As mentioned in the previous chapter, we would also like to determine some basis that can help determine how the network is able to classify a whole slide image as tumor or normal. Here, two approaches are proposed. The first involves a visual test on the distribution of tiles classified as tumor or normal in the whole slide. Following the process described in Fig. 3.3, a heatmap can be generated to have a picture of the scattering of tumor tiles. An example is shown below:

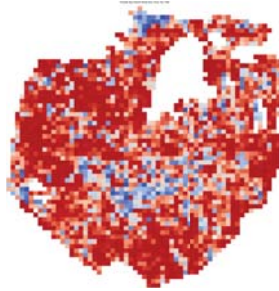


Figure 3.4: Image heatmap of whole slide

Fig. 3.4 shows the heatmap for the whole slide image following Fig. 3.3. Red signifies tumor, while blue signifies normal.

The second approach involves a comparison involving Table 3.1. We can then check the values for each column and determine some connection between these values and the classification.

3.5 Connecting Image Features to Genomic Features

We want to determine any correlation between image features gathered from whole slide analysis to genomic features such as mutation status and RNA-Seq values.

- For mutation status, we gather driver genes for prostate adenocarcinoma (as described in [34]), and check samples whose driver genes are mutated
- For RNA-Seq, we use HTSeq reads (60483 transcripts per sample) from Genomic Data Commons.

To make sure that we only consider relevant transcripts, and as a way to perform some dimensionality reduction, we reduce the 60483 transcripts by filtering those whose percentage of gene expression counts fall below some threshold. As an example, we look at the RNA-Seq data for 551 TCGA-PRAD cases. We dictate the minimum inclusion percentage count to be 70%, and the minimum log value to be -20. We see the result in Fig. 3.5.

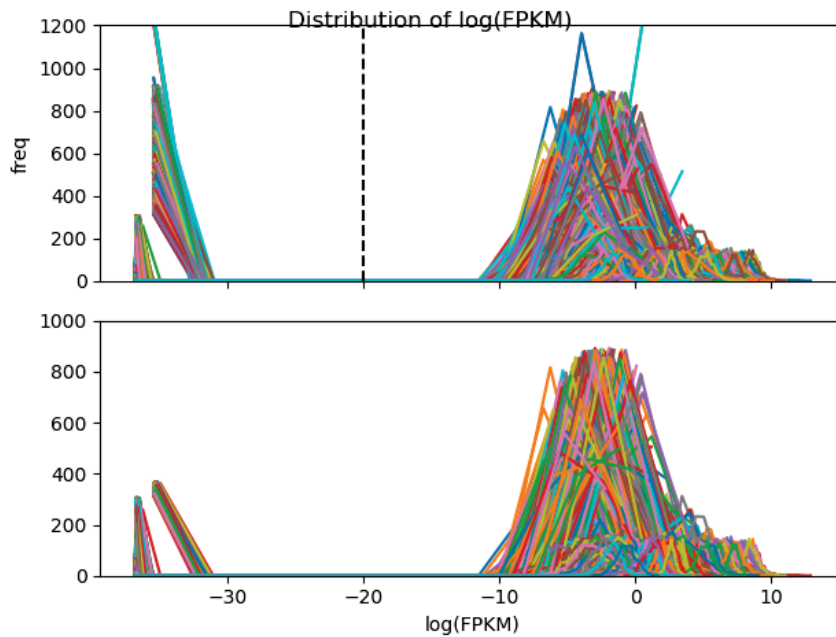


Figure 3.5: (Top) original histogram for RNA-Seq Data (Bottom) Histogram for remaining transcripts

We present and implement two methods for connecting image data with genetic information whose details are summarized below:

1. We implement a linear model whose details are given below.

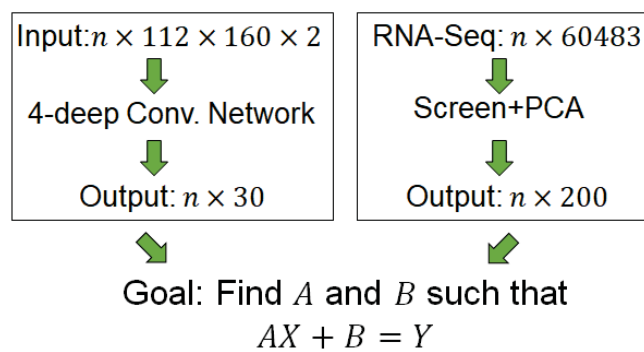


Figure 3.6: Linear model pipeline

2. We reduce the dimension of the image features using t -distributed stochastic

neighbor embedding (t -SNE) and use the reduced information for mutation status analysis.

Prostate adenocarcinoma (PRAD) whole slide images from The Cancer Genome Atlas (TCGA) database were gathered. Because of their size, these images were tiled into 512x512 subregions and used as input for the Inception-v3 architecture. The features were extracted based on a tumor-normal classification. The correct labels for the tiles were defined by some predefined threshold which depended on the tumor percent annotation from TCGA.

The probabilities from the softmax layer of this network were then collated and used as input for a multi-stage convolutional network; each stage is composed of convolutional, pooling, and normalization layers. A fully-connected and a softmax layer were then attached to the final stage and the features were learned based on a tumor-normal-metastatic classification.

For the linear model, HTSeq-FPKM data was used for quantifying gene expression. Each patient has expression values for 60483 transcripts. Data from patients in the TCGA-PRAD study were collated, and these reads were screened to remove non-essential transcripts as defined by some minimum threshold. A principal component analysis was performed on the remaining values to reduce the dimension to 200 principal components (PCs). A linear model was then carried out to associate the features extracted from the fully-connected layer of the CNN pipeline to the 200 PCs.

The second method that we applied to connect the image feature t -SNE to mutation status is described as follows:

From the t -SNE plot, we manually choose regions of interest that has a significant visual characteristic like morphology. Then, for each whole slide, we count the number of tiles belonging to that slide in each cluster. Because there is a huge variance in the number of tiles per slide, we make the data uniform by turning them into percentages.

In general, if we have manually chosen k clusters, then each whole slide will now be represented by a k -dimensional vector from which analysis can be performed.

Example 10. Here we have two patients. If we choose to have k clusters, then we will have a total of $k + 1$ groups (the last group represents the region outside

of our manually chosen clusters).

Patient	0	1	...	k	Total
1	15	25	...	90	180
2	26	8	...	66	120

Patient	0	1	...	k
1	0.083	0.139	...	0.5
2	0.217	0.067	...	0.55

4 Results and Discussion

We first verify the effect of having convolution on the accuracy of the classifier. The results are shown in Table 4.1.

It can be seen here that performing convolution before the softmax layer enhances the accuracy of the classification. This has to do with the fact that performing convolutions enables the emergence of patterns that the standard RGB input may not describe.

Table 4.1: Confusion Matrices for Varying Networks

NO Convolution					
subtype	Prediction			Total	Accuracy (%)
	PI	PP	TRU		
PI	16	0	34	50	32
PP	21	19	10	50	38
TRU	3	0	47	50	98
Total	40	19	91	150	54.7

Convolution Only					
subtype	Prediction			Total	Accuracy (%)
	PI	PP	TRU		
PI	32	0	18	50	64
PP	14	35	1	50	70
TRU	1	0	49	50	98
Total	47	35	68	150	77.3

Mean-pooling CNN					
subtype	Prediction			Total	Accuracy (%)
	PI	PP	TRU		
PI	26	2	22	50	52
PP	8	41	1	50	82
TRU	3	0	47	50	94
Total	37	43	70	150	76.0

Max-pooling CNN					
subtype	Prediction			Total	Accuracy (%)
	PI	PP	TRU		
PI	43	0	7	50	86
PP	14	33	3	50	66
TRU	0	0	50	50	100
Total	57	33	60	150	84.0

Now we look at the performance of the RISA network under similar conditions. The results are shown in Table 4.2.

Table 4.2: Confusion Matrix for RISA

RISA					
subtype	Prediction			Total	Accuracy (%)
	PI	PP	TRU		
PI	20	2	28	50	40
PP	25	17	8	50	34
TRU	0	0	50	50	100
Total	45	19	86	150	58.0

4.1 Visualization of Filters

We then look at the results of the reconstruction algorithm. While the actual slides are paired with their respective transcriptome subtypes, we use the unlabeled tiles for the autoencoder, and apply the labeling for the classifier. Now, we trained three stages of autoencoder as pretraining. Fig. 4.1 shows an example of the output of the first stage of the autoencoder. The original images here come from the general collection of images.

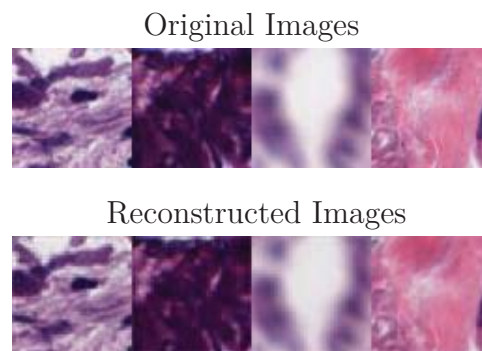


Figure 4.1: Example of the output of the autoencoder

We now look into some of the activations of the autoencoder. Though some color hue changed after reconstruction, the structural detail of the original input

was recovered from compressed information of encoded layers whose resolution is one fourth of the original image, as shown in Fig. 4.2.

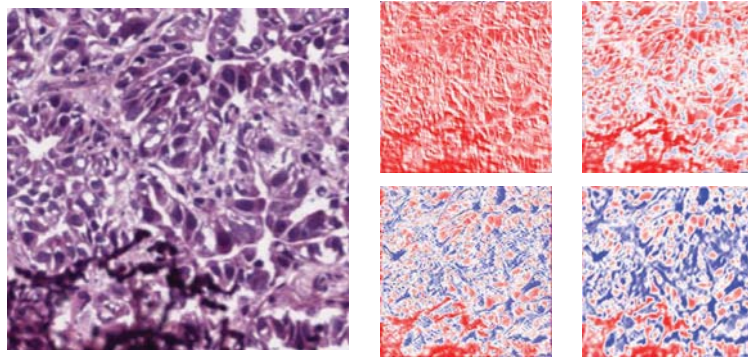


Figure 4.2: Left: input image, Right: output of some encoding layers in the second autoencoder. The gradient from red to blue represents increase in signal intensity.

In order to understand how the network extracts features after training, we randomly clipped the original images to generate 10000 sample input of size 32×32 pixels. Then, we computed the output of the encoding layer and sorted them according to the value of one node in the encoding layer of the third stage. The goal of Fig. 4.3 is to emphasize a specific feature extracted by the autoencoder. We take the average of the pixel intensities of the top 100 encoded images based on the sorted feature activation. A sample image is then obtained representing the activation in one of the encoding layers. This represents a feature of the training image patterns. It seems that the figure exhibits different local structures of cell boundaries such as stripe- or target-like patterns.

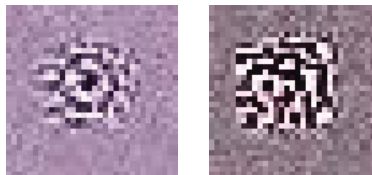


Figure 4.3: Examples of optimized local image for encoded outputs

4.2 Internetwork Comparison

We want to determine if the convolutional filter size has an effect on the reconstruction outputs and the classification accuracy of the networks.

For the following experiments, we used 64×64 images as input, and the networks follow the pipeline described in Fig. 3.1. First, we take a look at the reconstruction. Here we vary the convolution filter size on a standard convolutional autoencoder and a RISA network.

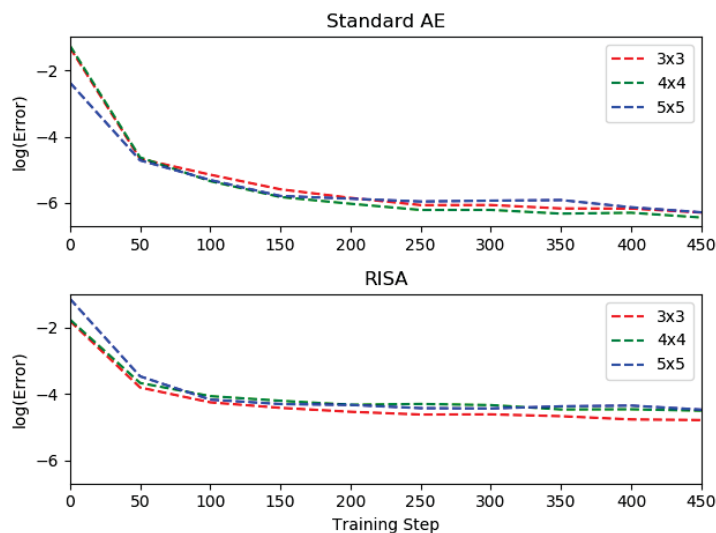


Figure 4.4: Comparison of AE and RISA training.

The results are shown in Fig. 4.4. We take the natural logarithm of the reconstruction error over the number of training steps. Here, 3×3 , 4×4 , and 5×5 are the convolutional window sizes. It can be observed that performance does not vary significantly as we change the filter size. However, it can be seen, especially in the RISA network experiment, that a slight increase in reconstruction performance is brought about by a decrease in the filter size. This implies that a smaller receptive field works better for this type of task.

Next, we performed a comparison between the activated filters of each of the networks that we are working on. We take the activations of the first layer of the direct classifier, the first stage of the autoencoder, and the lone convolutional layer of the RISA network. The goal here is to determine and hopefully interpret


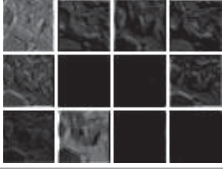
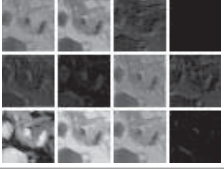




	Direct Classifier	AE	RISA
Original			
Filters			
Reconstruction			

Figure 4.5: Comparison Between Filters and Output of Networks

the features extracted from each of the networks.

In Fig. 4.5, we can observe several differences in the types of filters extracted. We observe that the output for the direct classifier shows some edge detection scheme through the contours in some of the filters. On the other hand, the standard autoencoder seems to emphasize shape and hue. The RISA network shows features similar to the standard autoencoder, but we also observe that some of them have paired up as part of the underlying architecture of RISA. (Note that the RISA filters were scaled to match the filters of the other networks.)

In the interest of finding the most accurate implementation of the convolutional classifier, we continue the experiment of varying the size of the input along with the convolutional filter size of the network variants. Specifically, we incorporate 32×32 , 64×64 , and 128×128 experiments. Table 4.3 summarize the accuracy of the different convolution models. In this table, we can see that, in general, there is some slight improvement in performance when we increase the filter window size. However, if we look at the accuracies of the RISA network, we see a different result. This can be attributed to the fact that as we increase the filter size, we have a relatively significant drop in reconstruction performance. It must be said that the effect doesn't seem to be drastic for the standard AE.

Table 4.3: Subtype classification accuracy tables for varying networks and filter sizes

(#) - number of test images

Window Size	32×32 (12000)		
	Direct Class.	AE+Class.	RISA+Class.
3×3	73.6	76.4	52.5
4×4	74.1	65.9	50.9
5×5	80.5	68.8	71.3

64×64 (3000)		
Direct Class.	AE+Class.	RISA+Class.
82.9	74.7	89.2
87.8	79.5	62.5
89.0	82.2	71.2

128×128 (750)		
Direct Class.	AE+Class.	RISA+Class.
68.4	73.3	56.7
86.4	74.3	35.9
89.1	54.9	72.1

4.3 Deeper Networks

Using a pretrained three-stage sparse autoencoder network, we trained to classify the transcriptome subtypes. First we confirmed the effect of block reduction. We evaluated the accuracy of the network by changing the input image size. This time, we used 128×128 , 512×512 and 2048×2048 images as input. From the results described in the previous section, we see that there is some advantage to altering the filter size of the autoencoder. As such, we use 7×7 , 5×5 , and 3×3 for the filter sizes of the three stages of the autoencoder, respectively, and 16×16 for the classifier.

To actually perform the classification on the 2048×2048 images, we first divide them into smaller tiles on which to apply the pre-trained convolutional autoencoder. We then concatenate the output of the final stage of the autoencoder and use it as input for the convolutional classifier.

Table 4.4 shows that when the input size was small, the network could not learn the difference between transcriptome subtypes very well. But as we increase the input size, more information is being read by the network, and hence, more complex features are extracted. Accordingly, the accuracy increases. It is worth noting that the number of nodes were not changed for the three experiments.

Table 4.4: Confusion matrices and accuracy for 128px, 512px, and 2048px experiments

128px						
prediction						
subtype		TRU	PP	PI	Total	Accuracy (%)
diagnosis	TRU	47	32	1	80	58.8
	PP	20	64	11	95	67.4
	PI	16	21	44	81	54.3
Total		83	117	56	256	60.5

512px						
prediction						
subtype		TRU	PP	PI	Total	Accuracy (%)
diagnosis	TRU	54	32	0	86	62.8
	PP	15	48	15	78	61.5
	PI	17	16	59	92	64.1
Total		86	96	74	256	62.9

1024						
prediction						
subtype		TRU	PP	PI	Total	Accuracy (%)
diagnosis	TRU	75	1	9	85	88.2
	PP	2	59	11	72	81.9
	PI	0	0	99	99	100.0
Total		77	60	119	256	91.0

2048px						
prediction						
subtype		TRU	PP	PI	Total	Accuracy (%)
diagnosis	TRU	60	0	0	60	100.0
	PP	0	49	1	50	98.0
	PI	1	0	65	66	98.5
Total		61	49	66	176	98.9

4.4 Whole Slide Image Analysis

This section is dedicated to our approach on connecting gene expression data to our image features. Here we use 704 TCGA-PRAD whole slide images consisting of normal, tumor, and metastatic images.

4.4.1 Classification of Whole Slides

Initial implementation suggests that the performance of the convolutional classifier fairs similarly to the mean probability taken from the tiles of each whole slide. We also introduce an analysis of decile threshold to determine if some threshold better classifies the images.

Further analysis shows that the convolutional classifier amplifies the mean probabilities gathered from the Inception architecture

The process of gathering the basic input for the whole slide convolutional network was presented in Fig. 3.3. The goal of the network is to classify the annotation of the image, with possible options being Solid Tissue Normal (STN), Primary Tumor (PT), and Metastatic (M).

Fig. 4.6 and Table 4.5 shows the training and testing performances of the network, respectively.

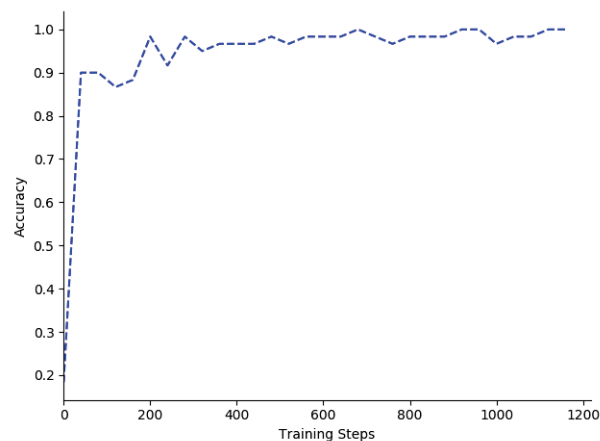


Figure 4.6: Training Accuracy for whole slide network

Table 4.5: Testing Accuracy for whole slide network

		prediction			Total	Acc. (%)
diag.		STN	PT	M		
	STN	133	87	0	220	66.5
	PT	19	1181	0	1200	98.4
	M	0	5	0	5	0.00
	Total	152	1273	0	1425	92.2

We now compare the performance with the mean inception probability as described in Table 3.1.

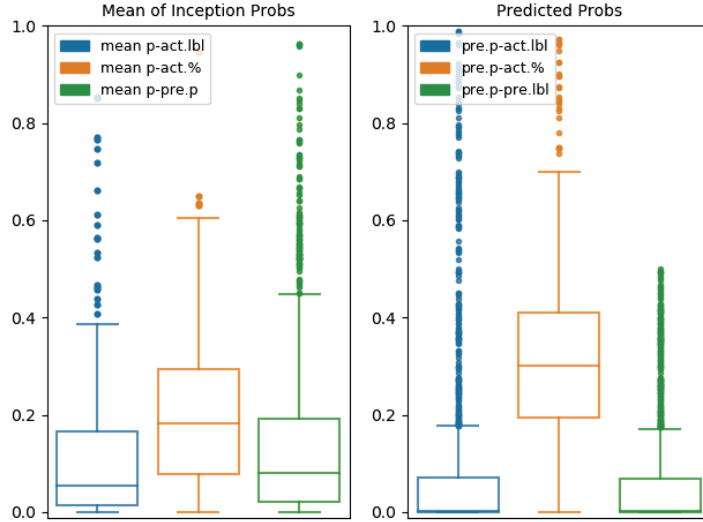


Figure 4.7: Error boxplots with Inception-v3 probabilities (left) and predicted probabilities (right)

The plots on the left of Fig. 4.7 involve mean probabilities obtained from the Inception-v3 architecture. The blue, orange, and green plots represent the differences between the mean probabilities and the actual slide labels, actual tumor percentages, and the predicted probabilities, respectively.

On the other hand, the plots on right focus on the probabilities obtained from the whole slide classification network. The blue, orange, and green plots represent

the differences between those probabilities and the actual labels, actual tumor percentages, and predicted labels, respectively.

One of the main observations that can be made here is the notion that the convolutional network amplifies whatever decision was made by the mean Inception probabilities. This is represented by the drop in the distributions of the blue boxplots. However, it can also be seen that the distribution increased for the error between the probabilities and the tumor percentages (orange plot). Both of these findings go hand-in-hand because the task of the network is to perform classification using features from Inception.

We then proceed with plotting ROC curves as another basis for comparison between Inception and the whole slide image network.

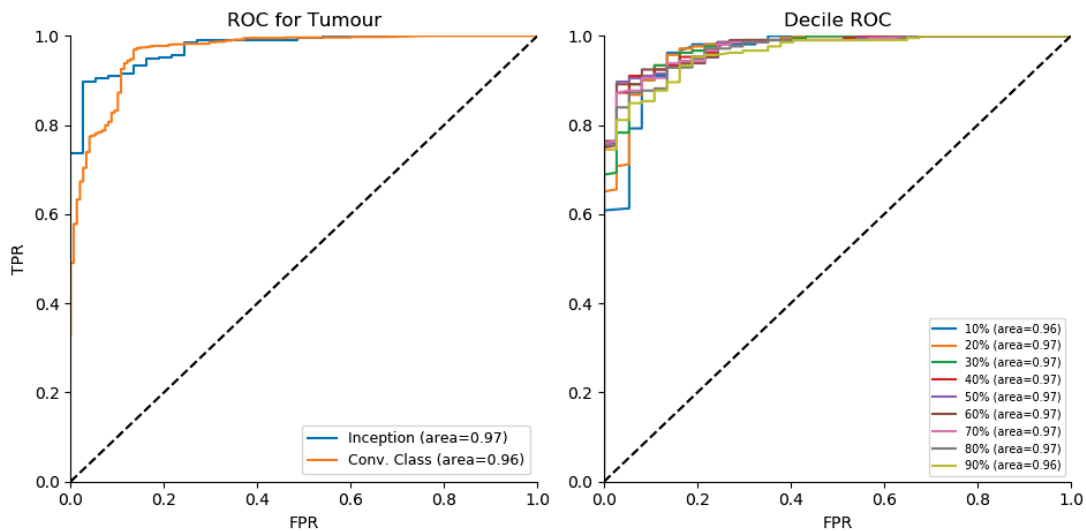


Figure 4.8: ROC Comparison

The left plot in Fig. 4.8 shows the comparison between the ROC curves obtained from the mean Inception probabilities and the convolutional classifier.

We also implement create ROC curves where several decile bounds were used. This is done in order to see whether there will be a critical decile bound that can be used for a more accurate classification. The plot on the right visualizes this experiment.

Overall, we see that the classification is decent, but further improvements to

the whole slide image network may be necessary.

4.4.2 Interpretation of Classified and Misclassified Slides

We now follow the full error computation analysis described in Table 3.1. From Table 4.6, we list the information for some slides and determine some details about the corresponding tumor percentage and mean tumor probabilities of correctly classified and misclassified slides. We can supplement this with the concentration of tumor images via a heatmap, as seen in Fig. 4.9. Essentially, this method can be used to determine whether images, despite having the same tumor percentage, might have different classification based on their concentration or density. In Fig. 4.9, for example, patients EJ-5502-01A-01-TS1 (left) and G9-6347-01A-01-TS1 (right) might have both been classified as PRAD because of some significant scattering of tumor cells, or some other similar characteristic. It is worth noting that EJ-5502-01A-01-TS1 was correctly classified, while G9-6347-01A-01-TS1 was not.

Table 4.6: Slide Information

Patient	Tumor %	Diag.	Pred.	Mean p	Pred. p
EJ-5502-01A-01-TS1	55	PRAD	PRAD	0.532	1.000
G9-6385-11A-01-TS1	None	NORM	NORM	0.307	0.212
G9-6347-01A-01-TS1	70	PRAD	NORM	0.161	0.0002
EJ-8474-11A-01-TS1	None	NORM	PRAD	0.512	0.998

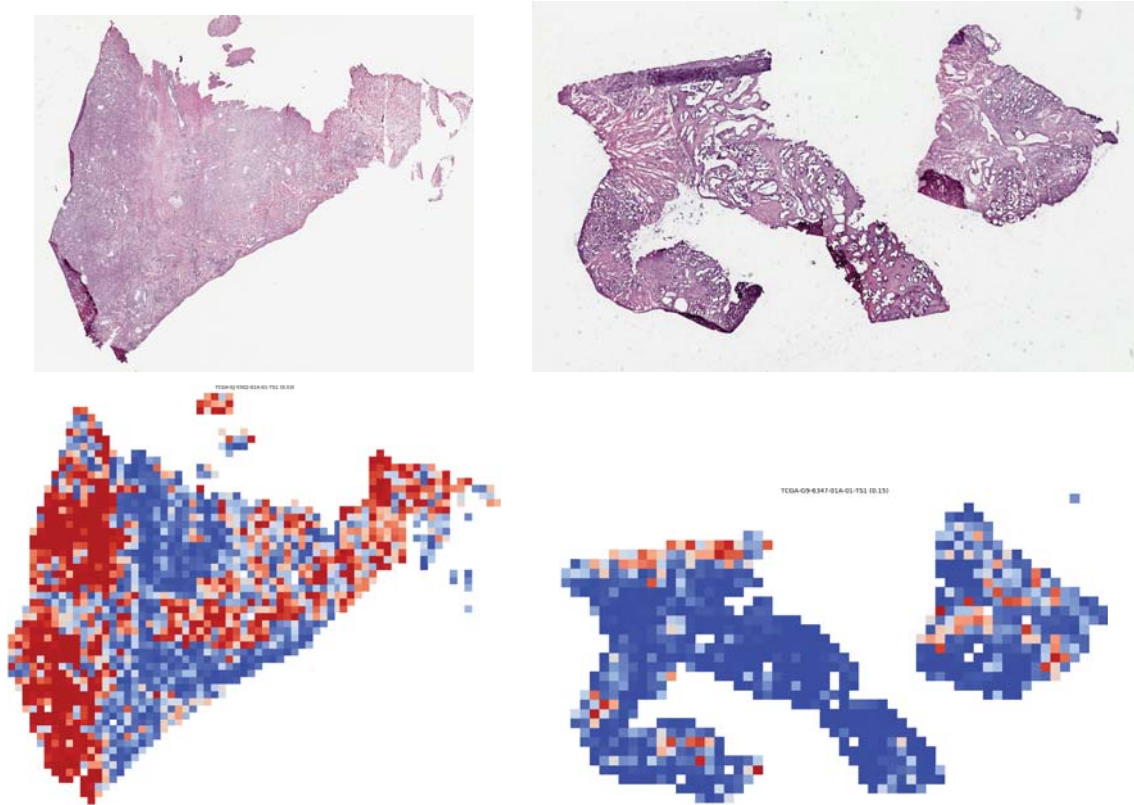


Figure 4.9: Whole Slide and Corresponding Probability Visualization

4.4.3 Connecting Image and Genomic Features

We want to determine any correlation between image features gathered from convolutional classifiers to genomic features such as mutation status and RNA-Seq values. Specifically, for mutation status, we gather driver genes for prostate adenocarcinoma, and check samples whose driver genes are mutated. For RNA-Seq, we use HTSeq reads (60483 transcripts per sample) from Genomic Data Commons (GDC).

We highlight the mutation status of several driver genes with the most frequent somatic mutations based from the GDC database. Ultimately, we want to be able to establish a correlation between the image and genomic features.

Fig. 4.10 visualizes the result of the linear model, whose process was described in Fig. 3.6. Each graph here is composed of the actual expression values of the

indicated gene, and the samples were separated based on whether the patients has that specific gene mutated or not. The value enclosed in the parentheses represents that p -value representing whether there's a difference between the mutated and non-mutated samples. The black line represents the mean value of the predicted gene expression from the linear model. Even though the overall performance of the linear model is poor (coefficient of determination=0.56), we can see that the prediction is decent by focusing on the driver genes for PRAD.

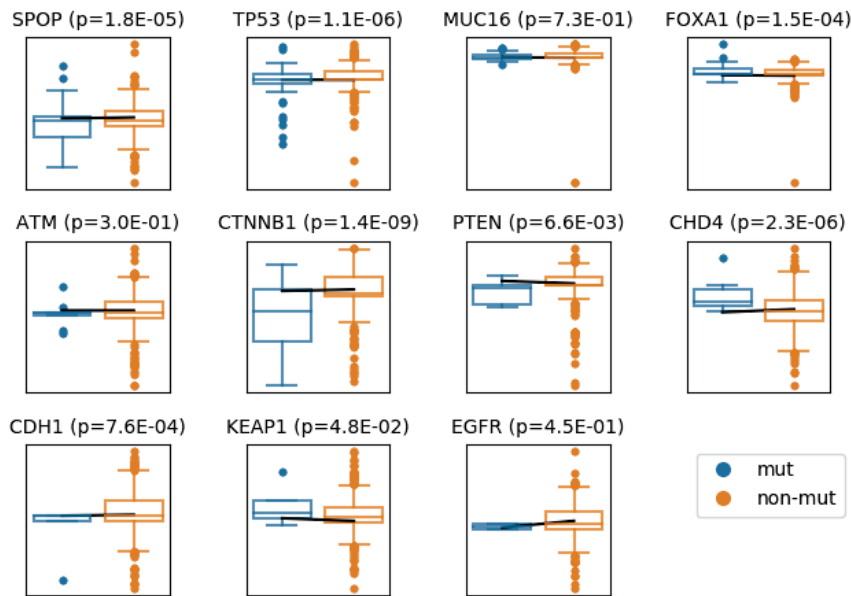


Figure 4.10: Gene expression boxplot with predicted expression overlay

Finally, we go back to the output of the fully-connected layer of Inception-v3. We perform t -SNE on the 2048-dimensional vectors to reduce it to two dimensions.

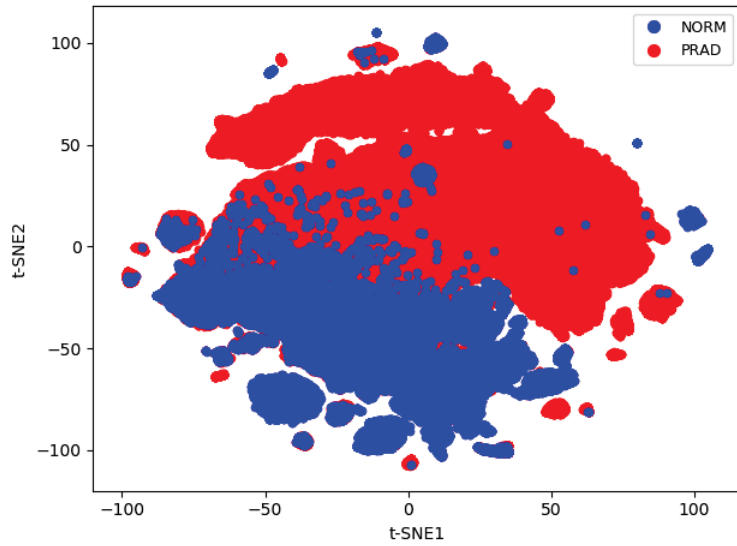


Figure 4.11: t -SNE plot for output of fully-connected layer

We see from Fig. 4.11 that there is a specific region where most of the normal tiles lie.

We now perform the analysis as described in **Exp. 10**. 8 clusters were manually chosen based on morphology. We see the result in Fig. 4.12.

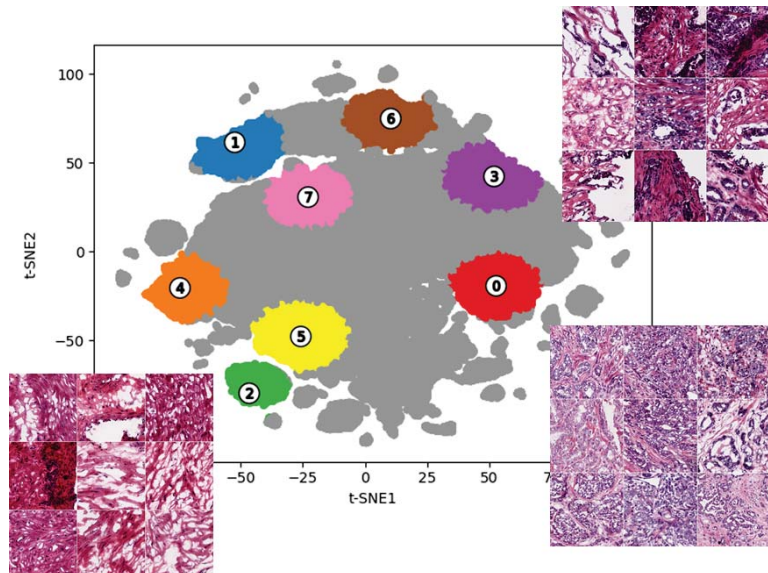
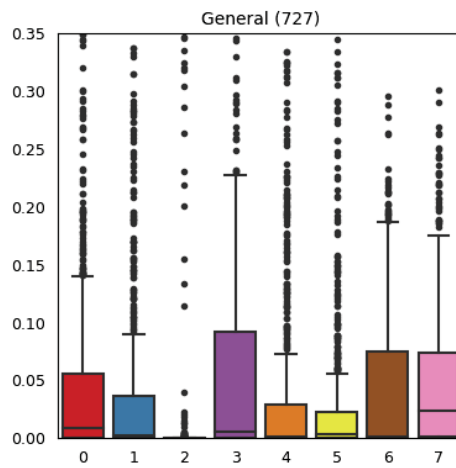


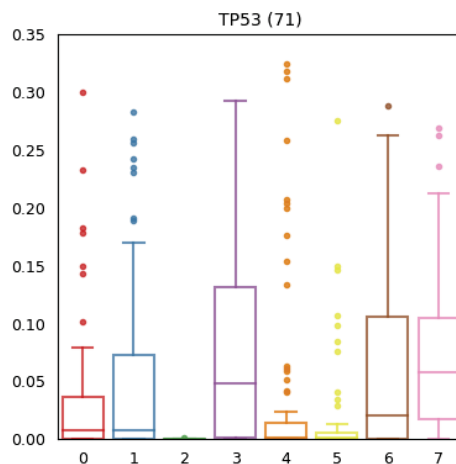
Figure 4.12: t -SNE plot with clusters and samples

The tiles on the lower left, upper right, and lower right portions of the figure represent a sample for clusters 2, 3, and 0, respectively. Cluster 0 seems to contain tiles with a high amount of dysplasia. On the other hand, cluster 2 is an island of normal tiles.

After computing the proportion vector for each sample, we visualize the proportions per cluster via boxplots, as shown in Fig. 4.13a. We can gather vital information with regard to mutation status by also focusing on some specific genes. We use TP53 here as an example. Samples which have a mutated TP53 gene were gathered. The boxplot for these samples are shown in Fig. 4.13b.



(a) Proportion boxplot for general dataset



(b) Proportion boxplot for TP53-mutated samples

The boxplots were then made for 15 driver genes. This time, we use driver genes based on [34].

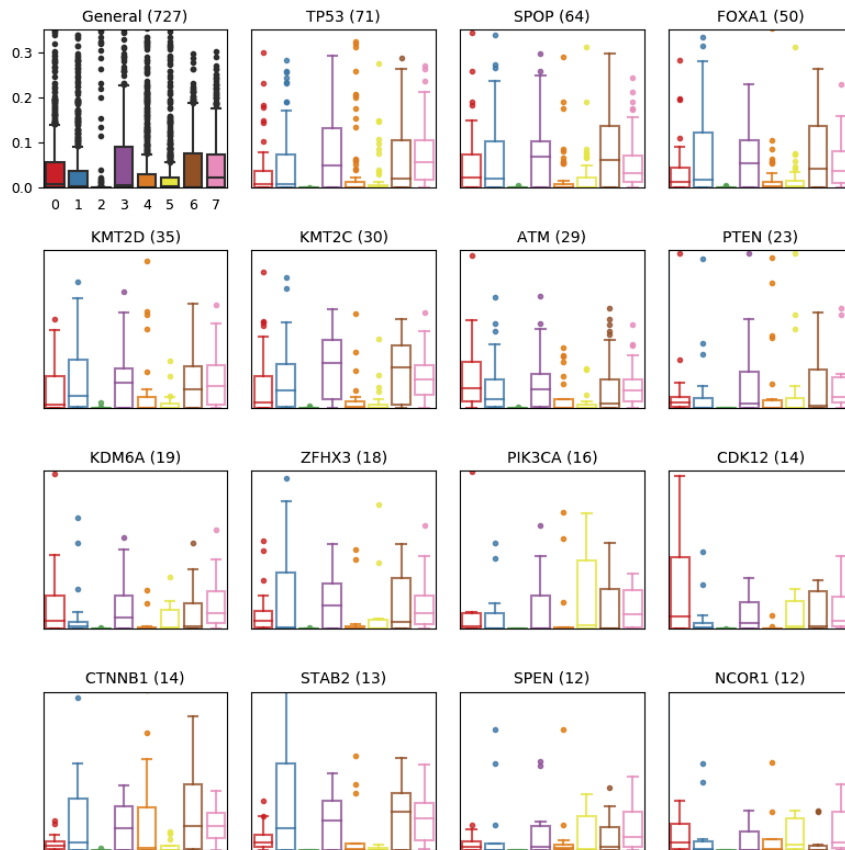


Figure 4.14: t -SNE boxplot for general dataset and 15 driver genes

Fig. 4.14 can be used as a reference for relating image features with mutation status. For example, if we look at PIK3CA, we can see that there is a significant visual difference between its boxplot and the general dataset, specifically for cluster 5. This tells us that cluster 5 contains a high number of tiles whose PIK3CA gene is mutated. The plot for SPOP is also worth noting because of the difference in the boxplots for clusters 1 and 6. This means that clusters 1 and 6 may contain a significantly high number of tiles whose SPOP gene is mutated. A more technical test should be applied to verify these notions.

5 Summary, Conclusions, and Recommendations

This study had two general goals. Firstly, we aimed to implement models involving CNNs for the reconstruction of lung adenocarcinoma images and the classification of transcriptome subtypes for this specific type of cancer. Secondly, we aimed to establish a connection between image features extracted from CNN architectures and genomic features.

The experiments using different input size indicate that the network requires a certain numbers of cells in the input images to recognize the difference between transcriptome subtypes. Looking at the differences of the convolutional filter output of each of the networks, we can see the features emphasized by the three variants. The direct convolutional classifier outperforms the other two networks and it can be seen that the important features relate to some combination of edge and hue detection. On the other hand, the autoencoder network emphasizes hue above all else. A deeper analysis of these filters is worth pursuing. Moreover, the pretraining implemented on the autoencoder-classifier networks provides several advantages like lower computational cost without a drastic effect on accuracy.

Using the pretrained autoencoder as a feature extraction mechanism for a convolutional classifier and tiling the 2048×2048 images into individual and independent tiles paved the way for a classification algorithm involving large image input, having a 98.89% test accuracy. Even though they belong to different clusters in gene expression profiles, it was difficult to distinguish them from their morphological phenotypes since their local cell structures were not so different. In order to distinguish statistical distribution of cellular features in larger tissue images, we introduced multiple reduction layers and succeeded to classify transcriptome subtypes correctly.

This new approach will be helpful for differentiation of various tissue types, not clearly different in cell morphology, but different in cellular distribution in the tissue. This result will help the diagnosis of lung cancer for appropriate treatment, and further applications will provide us useful tools for diagnosis of various tumor types.

We also implemented some analysis that could serve as a starting point for correlation with certain genomic datasets like gene expression profiles or mutation status. As such, information obtained from the whole slide image network was used for a number of experiments. One of the analyses involved the comparison of classified and misclassified whole slides. Having a deeper understanding of the mechanics of the classification could give us a better grasp of how the network performs the labeling and the specific features that it looks for in the images. In addition, a similar analysis could also be performed on the network with 2048px images as input.

The analysis following the results of the whole slide image model serves as a good starting point for outcome and prognostic predictions. Developments in the starting CNN architecture (Inception-v4, et al.) and the convolutional classifier can prove beneficial for connecting the image clusters formed by the overlays and those formed by gene expression. We also plan to implement the architecture to other classification schemes (like specific subtypes for certain cancers) and compare results with our current schemes.

Further improvements are necessary for whole slide classification. Some analysis can be performed to determine features and patterns to distinguish normal and tumor slides. Work still needs to be done to have a more decent correlation between image features and other established data.

Acknowledgements

Firstly, I would like to thank God almighty for giving me guidance and strength.

My family has been so patient while I finish my studies here, and for that I thank them.

I would like to express my deepest appreciation to the Computational Systems Biology laboratory of NaIST. Special thanks to Associate Professor Naoaki Ono and Professor Shigehiko Kanaya for their direction and counsel. In addition, I would also like to thank my committee members for providing valuable insight with regard to this thesis.

In addition, I would also like to thank the European Bioinformatics Institute (EMBL-EBI) for providing me an avenue to hone my skills as a researcher. Thank you to Moritz Gerstung for being a supportive and accommodating leader, and to my groupmates for being generous colleagues.

The results shown here are based upon data generated by the TCGA Research Network.

This degree would not have been possible without the Japanese government who funded my studies. I am also highly indebted to the International Affairs office for providing assistance whenever necessary.

I am also deeply grateful to the Filipino community in NaIST for providing moral support.

Finally, I would like to express my sincerest gratitude to Camille Marie Ruiz for accompanying me in this journey. I would not be where I am today without her loving support and encouragement, and for that, I am truly grateful.

References

- [1] Vijaya Karoor, Mysan Le, Daniel Merrick, Edward C Dempsey, and York E Miller. Vascular endothelial growth factor receptor 2-targeted chemoprevention of murine lung tumors. *Cancer Prevention Research*, pages 1940–6207, 2010.
- [2] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [3] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.
- [4] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [5] Hoo-Chang Shin, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Noguees, Jianhua Yao, Daniel Mollura, and Ronald M Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285–1298, 2016.
- [6] Yan Xu, Tao Mo, Qiwei Feng, Peilin Zhong, Maode Lai, I Eric, and Chao Chang. Deep learning of feature representation with multiple instance learning for medical image analysis. In *Acoustics, Speech and Signal Processing*

- (*ICASSP*), 2014 *IEEE International Conference on*, pages 1626–1630. IEEE, 2014.
- [7] Marc J Van De Vijver, Yudong D He, Laura J Van’t Veer, Hongyue Dai, Augustinus AM Hart, Dorien W Voskuil, George J Schreiber, Johannes L Peterse, Chris Roberts, Matthew J Marton, et al. A gene-expression signature as a predictor of survival in breast cancer. *New England Journal of Medicine*, 347(25):1999–2009, 2002.
- [8] William John Gradishar, Mark E Robson, Darl D Flake, Lee Steven Schwartzberg, Priyanka Sharma, Hatem Hussein Soliman, Anthony Martin Magliocco, Krystal Brown, Saskia Wehnelt, Alexander Gutin, et al. Predicting expected absolute chemotherapy treatment benefit in women with early-stage breast cancer using a 12-gene expression assay., 2018.
- [9] Moritz Gerstung, Andrea Pellagatti, Luca Malcovati, Aristoteles Giagounidis, Matteo G Della Porta, Martin Jädersten, Hamid Dolatshad, Amit Verma, Nicholas CP Cross, Paresh Vyas, et al. Combining gene mutation with gene expression data improves outcome prediction in myelodysplastic syndromes. *Nature communications*, 6:5901, 2015.
- [10] Ehsan Hosseini-Asl, Jacek M Zurada, and Olfa Nasraoui. Deep learning of part-based representation of data using sparse autoencoders with non-negativity constraints. *IEEE transactions on neural networks and learning systems*, 27(12):2486–2498, 2016.
- [11] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.
- [12] Q. V. Le, J. Han, J. W. Gray, P. T. Spellman, A. Borowsky, and B. Parvin. Learning invariant features of tumor signatures. In *2012 9th IEEE International Symposium on Biomedical Imaging (ISBI)*, pages 302–305, May 2012.
- [13] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, pages 52–59. Springer, 2011.

- [14] D. Neil Hayes, Stefano Monti, Giovanni Parmigiani, C. Blake Gilks, Katsuhiko Naoki, Arindam Bhattacharjee, Mark A. Socinski, Charles Perou, and Matthew Meyerson. Gene expression profiling reveals reproducible human lung adenocarcinoma subtypes in multiple independent patient cohorts. *Journal of Clinical Oncology*, 24(31):5079–5090, 2006. PMID: 17075127.
- [15] David Hayes, Carrie Lee, Patrick Roberts, Mary Beth Bell, Leigh Thorne, Jason Schallheim, Phil Bernard, and Bill Funkhouser. Subtypes of lung adenocarcinoma derived from gene expression patterns are recapitulated using a tissue microarray system and immunohistochemistry. *Cancer Research*, 67(9 Supplement):188–188, 2007.
- [16] Cancer Genome Atlas Research Network et al. Comprehensive molecular profiling of lung adenocarcinoma. *Nature*, 511(7511):543, 2014.
- [17] Matthew D Wilkerson, Xiaoying Yin, Vonn Walter, Ni Zhao, Christopher R Cabanski, Michele C Hayward, C Ryan Miller, Mark A Socinski, Alden M Parsons, Leigh B Thorne, et al. Differential pathogenesis of lung adenocarcinoma subtypes involving sequence mutations, copy number, chromosomal instability, and methylation. *PloS one*, 7(5):e36530, 2012.
- [18] Junji Shiraishi, Qiang Li, Daniel Appelbaum, and Kunio Doi. Computer-aided diagnosis and artificial intelligence in clinical imaging. In *Seminars in nuclear medicine*, volume 41, pages 449–462. Elsevier, 2011.
- [19] Maryellen L Giger, Nico Karssemeijer, and SG Armato. Computer-aided diagnosis in medical imaging. 2001.
- [20] Kunio Doi. Computer-aided diagnosis in medical imaging: historical review, current status and future potential. *Computerized medical imaging and graphics*, 31(4-5):198–211, 2007.
- [21] Jie-Zhi Cheng, Dong Ni, Yi-Hong Chou, Jing Qin, Chui-Mei Tiu, Yeun-Chung Chang, Chiun-Sheng Huang, Dinggang Shen, and Chung-Ming Chen. Computer-aided diagnosis with deep learning architecture: applications to breast lesions in us images and pulmonary nodules in ct scans. *Scientific reports*, 6:24454, 2016.

- [22] Paras Lakhani and Baskaran Sundaram. Deep learning at chest radiography: automated classification of pulmonary tuberculosis by using convolutional neural networks. *Radiology*, 284(2):574–582, 2017.
- [23] John Arevalo, Angel Cruz-Roa, Viviana Arias, Eduardo Romero, and Fabio A. González. An unsupervised feature learning framework for basal cell carcinoma image analysis. *Artificial Intelligence in Medicine*, 64(2):131 – 145, 2015.
- [24] Christopher D Malon and Eric Cosatto. Classification of mitotic figures with convolutional neural networks and seeded blob features. *Journal of pathology informatics*, 4, 2013.
- [25] Nandita Nayak, Hang Chang, Alexander Borowsky, Paul Spellman, and Bahram Parvin. Classification of tumor histopathology via sparse feature learning. In *Biomedical Imaging (ISBI), 2013 IEEE 10th International Symposium on*, pages 410–413. IEEE, 2013.
- [26] Holger R Roth, Le Lu, Jiamin Liu, Jianhua Yao, Ari Seff, Kevin Cherry, Lauren Kim, and Ronald M Summers. Improving computer-aided detection using convolutional neural networks and random view aggregation. *IEEE transactions on medical imaging*, 35(5):1170–1181, 2016.
- [27] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.
- [28] Hang Chang, Nandita Nayak, Paul T. Spellman, and Bahram Parvin. Characterization of tissue histopathology via predictive sparse decomposition and spatial pyramid matching. In Kensaku Mori, Ichiro Sakuma, Yoshinobu Sato, Christian Barillot, and Nassir Navab, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013*, pages 91–98, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [29] Jun Xu, Lei Xiang, Qingshan Liu, Hannah Gilmore, Jianzhong Wu, Jinghai Tang, and Anant Madabhushi. Stacked sparse autoencoder (ssae) for nuclei detection on breast cancer histopathology images. *IEEE transactions on medical imaging*, 35(1):119–130, 2016.

- [30] Sonal Kothari, John H Phan, Todd H Stokes, and May D Wang. Pathology imaging informatics for quantitative analysis of whole-slide images. *Journal of the American Medical Informatics Association*, 20(6):1099–1108, 2013.
- [31] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.
- [32] Jeremy T.-H. Chang, Yee Ming Lee, and R. Stephanie Huang. The impact of the cancer genome atlas on lung cancer. *Translational Research*, 166(6):568 – 585, 2015.
- [33] Kenneth Clark, Bruce Vendt, Kirk Smith, John Freymann, Justin Kirby, Paul Koppel, Stephen Moore, Stanley Phillips, David Maffitt, Michael Pringle, Lawrence Tarbox, and Fred Prior. The cancer imaging archive (tcia): Maintaining and operating a public information repository. *Journal of Digital Imaging*, 26(6):1045–1057, Dec 2013.
- [34] Joshua Armenia, Stephanie AM Wankowicz, David Liu, Jianjiong Gao, Ritika Kundra, Ed Reznik, Walid K Chatila, Debyani Chakravarty, G Celine Han, Ilsa Coleman, et al. The long tail of oncogenic drivers in prostate cancer. *Nature genetics*, 50(5):645, 2018.

Publication List

[1] Antonio, Victor Andrew A., et al. "Classification of lung adenocarcinoma transcriptome subtypes from pathological images using deep convolutional networks." *International Journal of Computer Assisted Radiology and Surgery* (2018): 1-9.