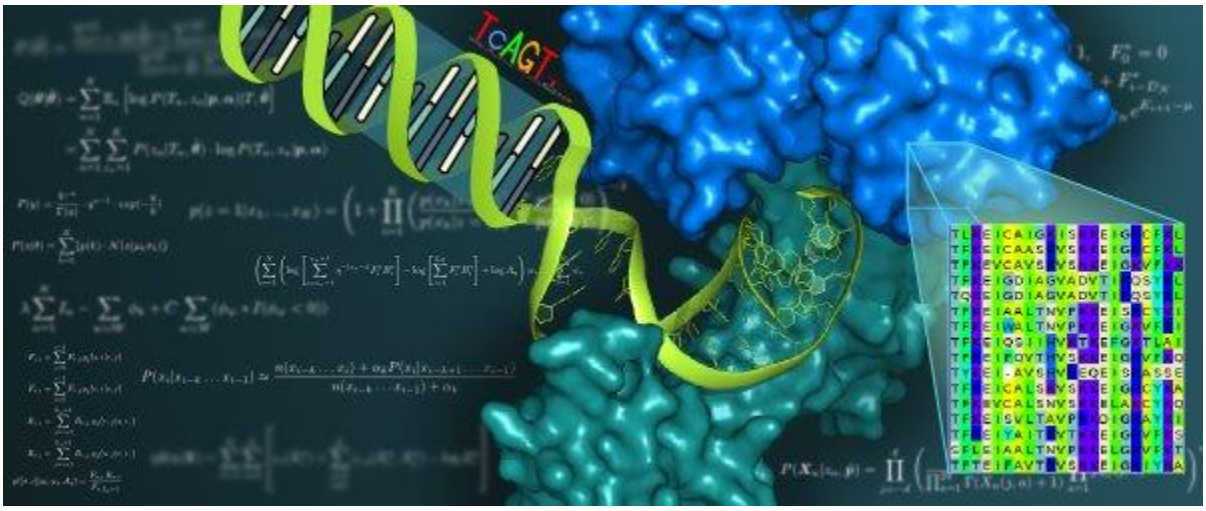
Deep Learning for Alternative Splicing

Team 17

Joe Kupferberg
Mark Liam Murphy
Nazih Bissat

Under the supervision of:

Dr. Yoseph Barash
Anupama Jha



II. Executive Summary

In this project, we set out to improve upon the state of the art in predictive computational models for alternative splicing: deep neural networks. Alternative splicing is central to human life and plays a role in a wide variety of human disease. It is of great interest to many researchers and medical practitioners alike; understanding the mechanism behind alternative splicing is key to truly unlocking the information in the human genome. With the advent of machine learning, splicing codes have become incredibly accurate in their predictions of how commonly a given exon is included in RNA and thus how frequently different proteins are synthesized. As these algorithms become increasingly accurate, the marginal improvements made upon them will ultimately come from new formulations of the data topology and a richer appreciation for the underlying biological phenomena. This semester, we have posited one such formulation, namely, that DNA is best thought of as an image and is therefore amenable to machine learning architectures that arose to perform image processing tasks. With this in mind, we have set out to create a Convolutional Neural Network to predict alternative splicing, and, although our initial results are lackluster, we believe that with more biologically tractable initializations, our innovative model will set the standard in computational biology.

III. Overview of Project

This semester, our team set out to accomplish a straight-forward goal: to build a splicing code that builds off of current state-of-the-art models and beats their accuracy and predictive power. Straight-forward as this goal may seem, the methodology entailed was anything but. To understand why this is the case, it is instructive first to understand alternative splicing, its importance in the biomedical sciences, and the splicing codes that aim to quantify it and against which we benchmark our project.

In a process known as gene expression, DNA is *transcribed* into RNA, which is then *translated*, or decoded to instruct the synthesis of proteins. DNA is made up of coding and non-coding segments known as exons and introns, respectively. During transcription, only exons are copied from DNA to RNA, as these are the segments of genetic material that ultimately instruct protein synthesis; introns are systematically excluded, or *spliced out* of the RNA sequence. Some exons are alternatively spliced; *i.e.*, they are not always included in the RNA sequence. As a result, a given portion of genetic code may give rise to multiple different proteins, depending on the inclusion or exclusion of certain exons. This process, which occurs at the level of transcription, is aptly dubbed *alternative splicing*. In this project, we limit our scope to the alternative splicing of cassette exons—exons flanked by two others, of which only the middle is alternatively spliced. This type of splicing is extremely common across species

Alternative splicing accounts for a tremendous amount of biological diversity, as it allows for the efficient storage of information in the genetic code. In fact, upwards of 95% of all human genes undergo alternative splicing. However, as is expected in a phenomenon as fundamental and common as alternative splicing, its misregulation can have extremely deleterious effects on the body and, in many cases, leads to disease. It is, therefore, of extreme interest to researchers to understand the mechanisms underpinning alternative splicing.

Splicing codes are mathematical models that in some sense quantify the process of alternative splicing. In particular, our model attempts to predict the frequency with which a given exon will be included in transcribed RNA, which is an excellent predictor of the type of protein to be synthesized. Prior to the development of splicing codes, biologists relied upon case-by-case analyses of a gene and posited cellular-, gene-, and tissue-specific factors to explain the splicing mechanism. The tremendous innovation in splicing codes was their ability to learn these factors computationally, without the limitations of human bias implicit in narrative-based logic. A seminal paper of Barash, *et. al.*, showed that probabilistic graphical models were particularly well suited to this task, and could predict the percent inclusion of a particular exon across different tissues in mouse genes with high accuracy. What's more, the data used was merely sequences of DNA labelled by tissue; the model learned the splicing mechanism entirely mathematically. Building on this work, Jha, *et al.*, drastically improved the predictive power by using a deep neural network architecture rather than probabilistic graphical models, implying that deep learning does indeed extract the necessary features from the data.

Our work builds off of the Jha model with two simple observation: DNA is an inherently spatial datum, and this spatial component is clearly linked to the splicing mechanism. As a result, the team hypothesized that convolutional neural networks, a learning architecture that arose from image processing, would be well-suited to capture the geometry of the problem and learn higher-order features to abstract the splicing mechanism.

If successful, a convolutional neural network with extremely high accuracy has the potential not only to make incremental improvements to biomedical research, but also clears the path for inference: biologists can use the results of our model to hypothesize about systems-level commonalities between

different splicing processes and extract a greater deal of information than was previously possible with hand-picked data or less sophisticated deep learning architectures.

IV. Method of Solution

Specification and Requirements. The proposed solution by our team for building an improved splicing code is to design, train, and evaluate a convolutional neural network. Past research by Dr. Yoseph Barash and his lab has shown deep learning to be an effective means for predicting alternative splicing events. A convolutional neural network relies on the same mathematical underpinnings as used in prior research, but utilizes convolutional filters to incorporate spatial elements of RNA sequences at splice sites and increase data richness.

The initial data is structured into six channels (**Figure 1**): one binary channel for each nucleotide, one channel for the "binding affinity" of a given sequence, and one channel representing the conservation of a sequence across species. These features were chosen for the initial model because they have been highly predictive in previous models. Two convolutional layers learn features from the channels, and these features are then concatenated with data indicating the tissue pair and used in a dense neural network to predict three outputs for a given cassette exon (**Figure 2**):

1. Percent inclusion in a given tissue.
2. Difference in percent inclusion for a given tissue pair.
3. Difference in percent exclusion for a given tissue pair.

These outputs are analyzed using conventional metrics: percent of variance explained (R-squared) and area under the receiver operating characteristic curve (AUC). Thus, for every pair of tissues, we compute R-squared from the predictions of percent inclusion of an exon, AUC for the classification of differential inclusion events, AUC for the classification of differential exclusion events, AUC for the classification of no change events. These metrics were used to evaluate past iterations of splicing codes, and so we have a clear comparison of every model we design to the models we want to beat.

Each model is trained using mini-batches of data, at first only containing events with high change in percentage inclusion across tissues, and then incorporating both changing and unchanging events as the training progresses. These models are trained using a Bayesian hyperparameter search to help choose many components such as learning rates, the number of convolutional filters in each convolutional layer, and the number of hidden units in each dense layer from an expansive hyperparameter search space.

Dependencies. This project relies heavily on applied mathematics and computer science skills and knowledge. A background in math combined with machine learning techniques is important for understanding the models and their mechanisms. Programming is essential for implementing and changing elements of models in Python. Working in a Linux environment and with version control software is another important component of the work, which we have had to largely pick up along the way.

The following classes are courses taken by members of our team which have been foundational for this project:

- ESE 224, 301, 302, 303, 305, 504, 530
- MATH 114, 240, 312
- CIS 120, 121, 519, 520

Also important to this project is a knowledge of microbiology and its terminology. Motivation for solving this problem is driven by an understanding of the underlying biological processes and the problems that face researchers trying to understand alternative splicing. The data used is derived from various biological

experiments, and reading past papers and using new datasets relies on an understanding of these experiments.

V. Self-Learning

Since we chose to take a project that was proposed on the M&T list, we have been working with a lab that has already done a lot of prior research on the prediction of alternative splicing. The first component that we had to self-teach was the mathematical formulation behind the lab's previous models. We had to understand the model's inputs, outputs, architecture, and loss function. Along with this, we read review papers on alternative splicing and learned about microbiology terminology so that we could understand the biological components of the project. We managed to grasp the main elements of the model as per our advisors' demands and we then shifted focus to the technical aspect of the project. To begin coding our model, all of the group members had to learn to use Linux commands and remotely log in to the lab's computer system, to use a version control system, to code in Tensorflow, and to work with Spearmint (the Bayesian optimization package we use for cross-validation and hyperparameter tuning). We used the first month of the semester to get accustomed to working remotely on the lab's system and running programs. The hardest component of the project that we had to self-teach ourselves was Tensorflow. It took us time to learn to use Tensorflow, and we began our learning process as a group. Once we all understood the basics of Tensorflow and the logic behind the lab's previous model using Deep Neural Networks, we were able to separate and individually learn some of the more advanced aspects of Tensorflow. At this point in the semester, we have a solid base of knowledge in Tensorflow and have been able to write and modify code for our model.

VI. Design and Iteration

Base Model

The base model was the dense neural network detailed in Jha, et al. This model used CLIP-Seq and RNA-Seq data to predict the percentage of splicing of cassette exons directly. The architecture consisted of an autoencoder layer and two dense layers.

Initial Convolutional Neural Network

The initial design that we dealt with was a convolutional neural network. The data was given as channel data for each cassette exon. This consisted of two convolutional layers and a densely connected network with two hidden layers, using the same target function as the dense neural network. This model only dealt with anonymized and IACUC compliant mouse data, and so complied with data standards. Software best practices were exercised in sharing and collaborating on code by using version control software.

Bayesian Hyperparameter Search

In order to explore a massive hyperparameter search space, we ran our initial CNN using Spearmint, a package for guided hyperparameter search. This software is open source, and did not change the implementation of the model itself. Its incorporation imposed no additional standards on our design.

Incorporation of Prior Knowledge

In addition to a massive hyperparameter search space, our model has a very large parameter search space. This is particularly true of the convolutional layers. As a result, we wanted to guide the model's learning towards filters of biological significance. One way of doing this was to incorporate Position Specific Scoring Matrices, or PSSMs, into the model, which are basic bioinformatic objects that indicate the likelihood of a given location's being a splice site or a binding site for regulatory proteins. As such, we initialized half of our filters at known PSSMs to help guide the learning.

Normalization Schema

Normalizing data can be integral to separating signal from noise in deep learning. In an attempt to get a decreasing training error across epochs, we attempted models with the following normalization schema:

1. Mean-centering (by channel per image)
2. Mean-centering and standard deviation scaling (by channel per image)
3. Mean-centering (by channel by dataset)
4. Mean-centering and standard deviation scaling (by channel by dataset)
5. Stretching to the interval [0,1]
6. Stretching to the interval [-1,1]

These normalizations did not impact the standards for the project, as they just change the data pipeline of the project

VII. Societal, Global and/or Economic Impact.

Between 90% and 95% of multi-exon human genes are alternatively spliced. Many human diseases are caused by the expression or non-expression of certain proteins, and the misregulation of alternative splicing has been proven to cause or modify diseases. Because of its prevalence in human disease, accurate prediction and deep understanding of alternative splicing is of interest to stakeholders across the biomedical sciences. Researchers interested in understanding a biological problem turn to splicing models for inference about splicing mechanisms; this is particularly important at the preclinical phase when researchers decide how much time and money to invest in a given project. Practitioners in personalized medicine could turn to splicing codes to predict splicing in a patient and craft individualized treatment plans based thereon. As the personalized medicine industry grows, companies will pay a premium for splicing codes with higher accuracy rates to secure competitive advantages. In the future, once models like ours are established, they can potentially aid with the detection of diseases.

We have two general categories of ethical concerns should our model undergo further development with human data and eventual commercialization. At the level of data collection, we want to avoid bias; we can imagine a scenario in which the data is collected on, say, one particular race or gender of people, and as such does not generalize well to racial minorities. This has profound implications in terms of discriminative or asymmetric quality of healthcare for different groups, and is something to be conscious of. A second concern occurs at the phase of commercialization; we foresee a scenario in which our data is used by insurance companies who want to price-discriminate. If they have data about potential policyholders' genetic code and are able to predict splicing aberrations that cause disease, they can use this as a nefarious rent-seeking tactic. We would therefore have stringent policies about licensure of the model and develop a business plan to help combat this issue, perhaps charging insurance companies proportionately to the benefits they get from this type of discrimination, making it infeasible.

VIII. Summary of Meetings

(Sept 14) Initial meeting with Dr. Barash

Attendees: Liam, Joe, Nazih, Dr. Barash

Summary: Dr. Barash explained his work and projects he had that he thought may be of interest to the team. The team explained their skills and background.

(Sept 19) Initial meeting with Anupama

Attendees: Liam, Joe, Dr. Barash, Anupama, Deep

Summary: Anupama, a PhD candidate with extensive computational biology experience, explained underlying biology of models and goals for the project. Deep, a new PhD student in the lab, sat in to learn more about Anupama's work.

(Sept 27) Technology setup meeting

Attendees: Liam, Joe, Nazih, Anupama, Jordi

Summary: Jordi, the individual in charge of IT for the lab, set up accounts for the team. Anupama answered more questions about the biology and formulation of the models.

(Oct 2) Review of existing model implementation

Attendees: Liam, Joe, Nazih, Anupama

Summary: The team reviewed the code from past models with Anupama and discussed developing in a virtual environment and training with Spearmint.

(Oct 7) Team meeting

Attendees: Joe, Nazih, Liam (remote)

Summary: Team meeting to touch base on training old deep network and develop using a virtual environment.

(Oct 23) Review of Spearmint, developing a CNN

Attendees: Joe, Nazih, Liam, Anupama, Deep

Summary: Anupama proposed her idea for a convolutional neural network and described the structure of the data.

(Nov 7) Implementation of CNN

Attendees: Liam, Joe, Nazih, Anupama

Summary: Team asked questions on the implementation of a CNN in Tensorflow.

(Nov 30) Evaluation of first model

Attendees: Liam, Joe, Nazih, Anupama

Summary: Team reviewed initial model results and discussed filter visualizations and the incorporation of PSSMs.

(Jan 16) Touch base for second semester

Attendees: Liam, Joe, Nazih, Anupama

Summary: First meeting after break, discussed second semester goals

(Feb 22) New dataset and design iterations

Attendees: Liam, Nazih, Anupama

Summary: Walk through new dataset Anupama prepared and additional design approaches

(March 20) Troubleshooting and debugging

Attendees: Liam, Anupama

Summary: Review models trained, normalization schema and other approaches, and training output.

(Apr 6) Review of final presentation materials

Attendees: Liam, Joe, Nazih, Anupama
Summary: Discussed wrap up for project, went over additional standards and considerations, reviewed methods tried, discussed future iterations of model.

IX. Final Schedule with Milestones

	Nazih	Liam	Joe
Complete Filter Visualization Pipeline	02/23		
Train and Evaluate Model on New Data		03/01	03/01
Implement Normalization Schema		03/14	
Debug Model			03/14
Train Improved Model	03/19		
Project Poster	04/07		

X. Discussion of Teamwork

Since none of the group members were familiar with Tensorflow, many of the tasks getting started were accomplished together. Our group members have communicated well over the course of the semester through our iMessage group, through GroupMe, and through emails. During the first half of the semester, we tried our best to meet several times a week and move forward together so that none of us would fall behind and miss important technical details. Once we were all familiar with the basic technical requirements and were able to run a Tensorflow program remotely on the lab's GPUs, we began to split the tasks up more and work individually on the project.

Nazih started implementing the model's convolutional layer in Tensorflow and the cross validation configuration file for Spearmint. Joe worked on implementing the model's convolutional layer. Liam worked primarily on running new parameter searches for the convolutional neural network and evaluating results. After realizing that our model's training error was not decreasing, Liam worked on implementing different normalization schema in order to improve this metric. Nazih implemented the pipeline to visualize the model's learned filters. Joe worked on debugging the model to find reasons/faults in its architecture that were causing training error to stagnate. Joe also researched the standards that apply to our project and made sure that we complied with them throughout this year.

XI. Budget and Justification

We originally stated that we had no budget for the project and this is still where we stand. We have been given access to the Biociphers lab's Graphical Processing Units to run our code and the packages we have needed throughout the semester have been free. Moving forward, we do not expect to need any money to complete our future tasks.

XII. Work for Second Semester

Over the next semester, we continued to iterate through the model. The model has done a relatively poor job of learning the way we anticipated it would. A large factor contributing to these lackluster results and their interpretation is the sheer size of the hyperparameter and parameter search space. The model is attempting to learn filters for the data; the sheer number of combinations it has to check is dumbfounding, and, therefore, the model's accuracy is highly sensitive to the initialization of filters. In fact, an active area of deep learning research inquires strategies for optimal initialization. As previously discussed, we employed a number of strategies (Bayesian hyperparameter search, incorporating prior knowledge, normalization schema, *etc*.) to try to guide the model's learning and improve accuracy.

This semester we also focused on filter visualization, as a way to help us understand and interpret the model's learning. With a working model, this pipeline will help future users of the model to answer the question of what, in fact, our model learns and what features it was able to extract as statistically significant components of the splicing mechanism. Practitioners can cross-reference these filters with the biology literature to try to understand if there is anything biologically interpretable about our results. The model's filters are meant to reduce dimensionality by extracting important features from our input data; visualizing the filters will thus show sequences of RNA that have the most significant impact on the model's output. Since deep learning models are complex, black box models, the filter visualization will enable us to infer genetic factors that influence alternative splicing and reverse-engineer the mechanism that regulates alternative splicing.

XIII. Standards and Compliance

Since our project is a bioinformatics project, there are two types of standards governing the project: on the hand, those surrounding healthcare data, and on the other, those governing software development best practices.

On the health care data side, we primarily considered three standards setting bodies, ranging from those that impacted us most to least immediately: IACUC, IRB, and HIPAA. The IACUC has no standards for usage of data post-collection; rather, it focuses on protocol and procedure for animal care and documentation at the point of collection. As a result, we ensured that the data we utilized in our model was IACUC compliant. The bulk of the data came from Keane et. al.'s 2011 "Mouse genomic variation and its effect on phenotype and gene regulation," a *Nature* paper that was itself IACUC compliant. Moving forward, should we begin to look at human data for the sake of greater generalizability of our model, Penn would require IRB compliance. To apply such a model to humans, we would need to anonymize our datasets. We can think of a couple of ways of achieving this; a simple stripping of identification fields would help to, though not entirely, solve this issue. To preclude the possibility of reverse-engineering characteristics of the subjects in the data set, we might consider end-to-end encryption of the data, or other cryptographic techniques for data-anonymization. Finally, our team

looked into HIPAA and its possible implications for our project. We found mostly that HIPAA standards would be subsumed by the IRB checklist. HIPAA applies to healthcare providers, health data clearinghouses, and health plans, none of which our lab constitutes.

On the computational side, we used software best practices such as version control, branched development, testing on subbranches, and then validating and merging to ensure our code ran well and to distribute the workload evenly and simultaneously. This strategy roughly corresponds to IEEE 12207.0, which governs software life cycle processes.

XIV. Conclusion

To date, we have successfully run deep learning models on genetic tissue data and tried different designs for convolutional neural networks with the help of our mentor. In addition to this, we learned a significant amount about the domain and tools we have began learning. We gained knowledge about deep learning, computational biology, and genetics as they relate to our project. We became comfortable developing in a Python virtual environment on a remote machine and learned about using version control software.

Future work on this model can improve accuracy by further tweaking its hyperparameters and iterating through the design process we discussed above. Moreover, a partial reworking of the mathematical and architectural framework could have positive results.

The most valuable takeaway to date has been how to apply our skills and knowledge in a discipline where we previously had no experience. Immersing ourselves in computational biology and deep learning has been overwhelming at times and has presented far more roadblocks than we could have imagined at the offset, but adapting to these challenges has come with opportunities for growth and the development of abundant transferable skills.

XV. Appendix

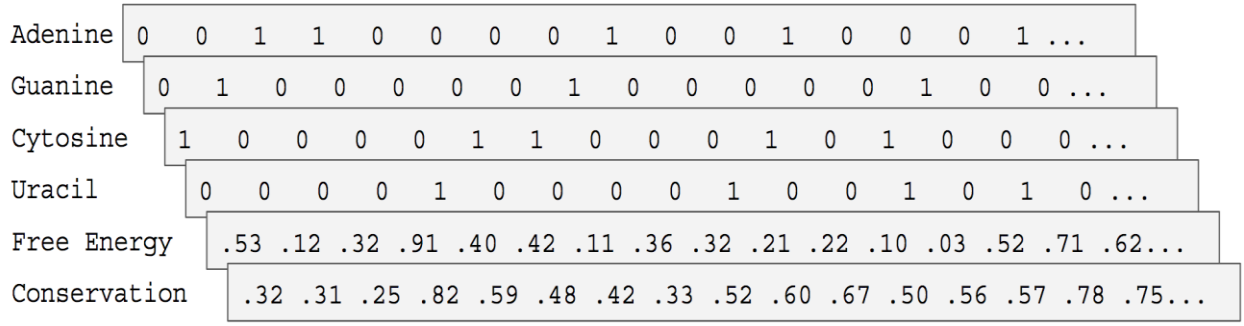


Figure 1. Six (6) channel data for the convolutional neural network. The first four (4) channels indicate which of the four nucleotides occurs at every spot. The data comes from read data that has depth of over 60 million, meaning that there is significant confidence for each nucleotide value. However, when there is uncertainty as to the value, the nucleotide values are a uniform probability distribution over the potential nucleotides. For instance, if it is unknown whether a given location is guanine or cytosine, the nucleotide channels will read (0, 0.5, 0.5, 0), as there is assumed to be an equal likelihood that the read is either guanine or cytosine. The fifth channel indicates the free energy of a binding site, which is a measure of the affinity of a given location to bind to another complex. One might expect that this increases the probability that a splieocome complex binds at a given location and, therefore, that a given portion of DNA is spliced out. Finally, the conservation channel indicates how common a genetic sequence is across different animals; conservation indicates a sequence's evolutionary benefits and suggests that a sequence is fundamental in its function. We might expect that sequences with higher conservation are less likely to be spliced, as these genes do not perform differential roles but rather serve on key purpose.

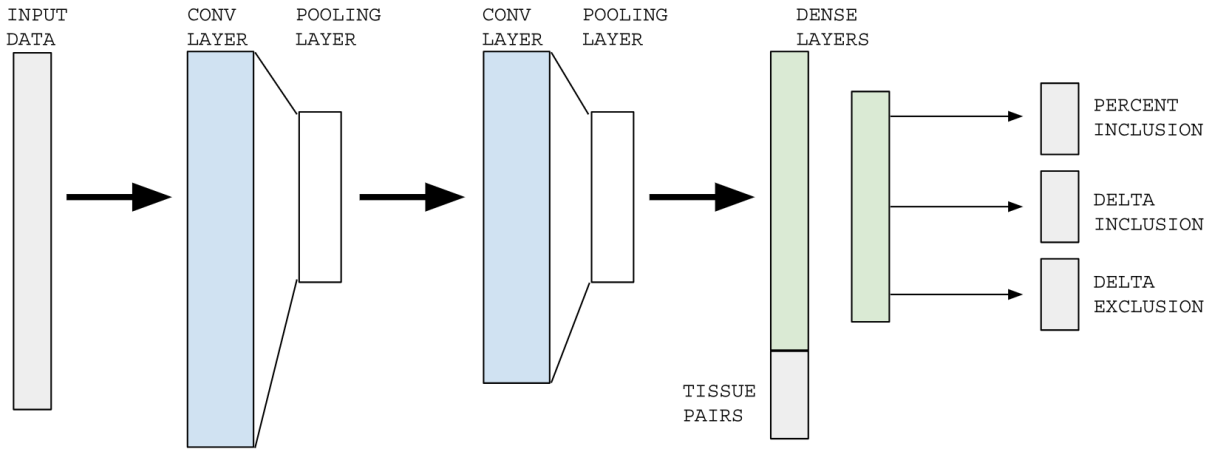


Figure 2. Initial design layout for a convolutional neural network. The input layer contains six channel sequences for every exon in the dataset. The convolutional layers consist of convolutional filters which learn features from the dataset and vary based on the hyperparameters. The output of the convolutional layers are then pooled to reduce dimensionality. These features are concatenated with one hot vectors encoding their tissue pairs, and then are trained on a dense network. The output layer predicts the percent inclusion of an alternatively spliced exon given its tissue and the difference in inclusion of an exon between every tissue pair.

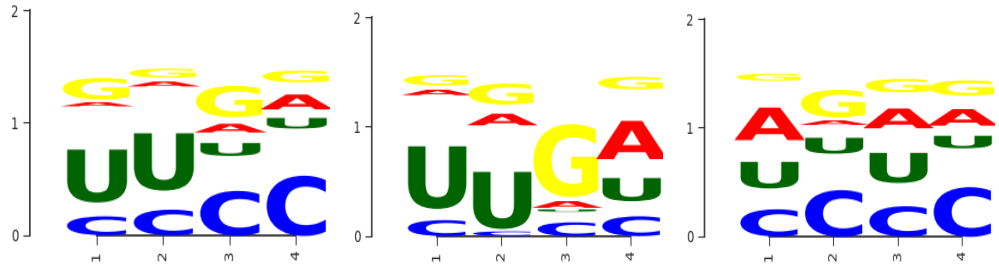


Figure 3. A batch of visualized filters. These filters perform the convolutional operator, and thus serve as feature extractors. The images above are standard bioinformatic visualizations of these kinds of filters, called Sequence Logos.