DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences

Team DanQ: Phila Dlamini, Jesalina Phan, Rosy Sun, Megan Carlson

Introduction

Paper: DanQ, Daniel Quang et al. 2016

Problem

- Build a model that predicts the function of non-coding DNA directly from sequence

What's novel

- Previous approaches relied solely on CNNs (e.g., DeepSEA), which are unable to capture long-range dependencies, or on SVMs (e.g., gkm-SVM), which require extensive feature engineering.
- DanQ uses CNNs to detect local motifs, and BiLSTMs to model the regulatory grammar between them

Why it's important

- Over 98% of the genome is non-coding, yet ~93% of disease-linked genetic variants are found in these regions

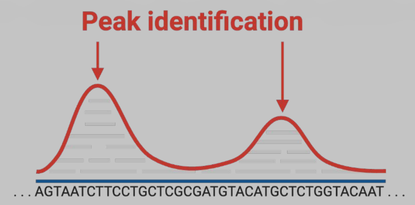
Data

The original paper was trained on the human GRCh37 reference genome using 919 unique transcription factors, which required a huge amount of compute power and time.

Instead, we decided to use **mouse data**, incorporating ChIP-seq data from 39 TFs. We pulled this data from the **ENCODE** project and the **UCSC genome browser.**

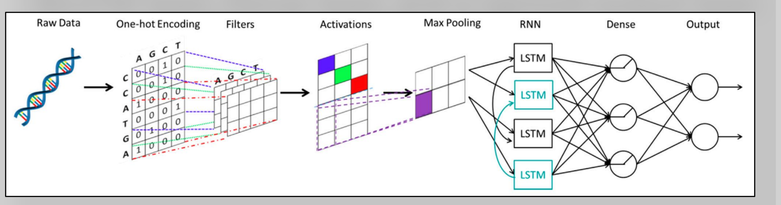
Architecture

Preprocessing: We split the mouse genome into **non-overlapping 200 bp bins** and overlapped them with **ChIP-seq peaks from 39 TFs**. Bins that contained at least one peak were kept and extended 400 bp upstream and downstream to generate reads of 1,000 bp. Labels consisted of 39x1 vectors representing the presence or absence of certain TF peaks in each bin.



[00010000000000...0000100000000000]

Model architecture: The model is a hybrid framework that combined **CNNS** and **BiLSTMS**. The model consists of 5 layers: A convolutional layer, max pool layer, BiLSTM layer, Dense layer, and a sigmoid output layer.

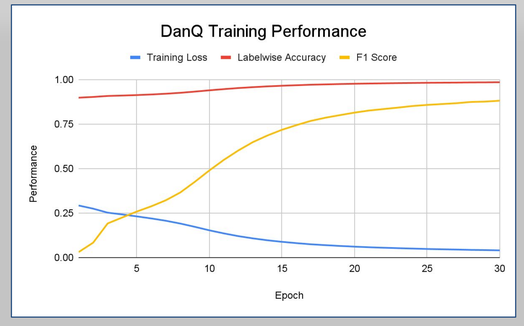


Issues

Preprocessing took a lot more computational power than expected. Our initial attempt totally wiped Phila's computer :(

Our model was able to train successfully, but it seems to have severely overfit the data. We suspect the overfitting arises from excessively complex architecture for our 39 transcription factors compared to the original paper's 919.

Results



- **Training accuracy starts really high** (in the low 90s), probably because of class imbalance – there are way more 0s in our target vectors than there are 1s. As such, **F1 scores are used to provide a more meaningful measure of the model's performance.**
- The model appears to be **sensitive to initialization.** In two separate training runs, F1 scores started out very differently—once as low as 0.003, and another time at a relatively higher 0.14. This was corroborated in the original paper, actually, which was the justification for DanQ-JASPER – a model initialized with known motifs.

DanQ Training Performance	
Average loss	0.041
Labelwise accuracy	0.985
F1 Score	0.881

DanQ Testing Performance	
Average loss	0.397
Labelwise accuracy	0.905
F1 Score	0.301

Model seems to be overfitting; F1 scores during training get really high (~0.85), but are much lower during testing (~0.3).

Future Directions

If we were to continue this project, we'd want to **address overfitting** with model simplification.

Given more time, we also would be interested in doing **model interpretation** to pull out which TF motifs the model is basing its functional predictions on. This kind of meta-analysis would give us greater insight into the biological role of each TF, which could be applied to wet lab research contexts.

Colorizer

Ben Silverman, Anoop Singh, Kshitij Sachan

Team Name: Heads Up! This queue is ending in five minutes!

Introduction

Paper: Colorful Image Colorization, Richard Zhang 2016

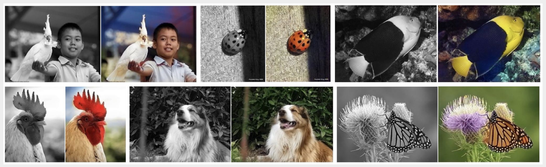


***Fig. 1**: Demonstration of paper's colorizer*

Problem:
Build a model that can generate plausible colorings for black and white images.

What's Novel:
- Previous image colorizing networks produce gray, desaturated images
- Desaturated images are caused when multiple colors are plausible for a given object and networks learn that gray minimize loss on average
- This paper uses a special loss function to create more plausible, colorful, colorizations

Why it's important:
- We can colorize legacy black and white images.
- Potentially useful for retouching images in the future
- Improve object classifiers

Data

Dataset: CIFAR-10

Lab Colorspace:
We converted our images into the Lab color space to be easier to work with
- 3 channels, L: Luminance (Black and White), A: Green to Red, B: Blue to Yellow
- L is input, (a, b) is output

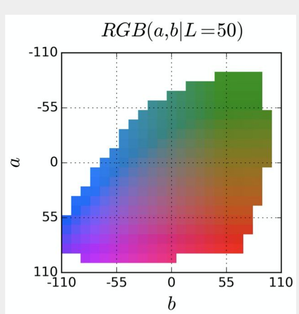


***Fig. 2**: The LAB color space is quantized into 400 bins*

Architecture

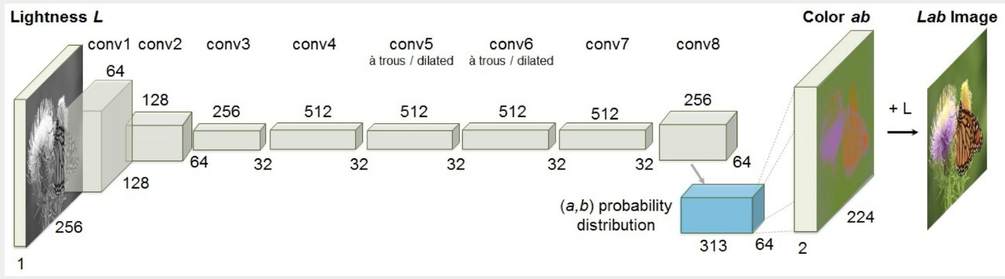


***Fig. 3**: Model architecture is 24 Convolution layers separated by ReLu and BatchNormalization*

Key Features:
- Input "image" has one channel (L) and output "image" has two channels (a, b)
- Sets of 2 or 3 convolution layers followed by batch normalization
- Transpose Convolution layers are used to scale up the image at the end
- **Bins:**
  - The a,b color space is divided into 400 bins that represent possible a,b combinations
  - Prior to training, a distribution of our label image's colors over these bins is calculated
  - The model first outputs probabilities over the 400 bins for each pixel of each image
  - Bin's represent some combination of a,b values used to calculate a final output
- **Special Loss Function:**
  - A multinomial cross entropy loss was used with reweighting to avoid images collapsing towards neutral colors
  - The loss was defined as: $\mathrm{L}_{cl}(\widehat{\mathbf{Z}}, \mathbf{Z}) = -\sum_{h,w} v(\mathbf{Z}_{h,w}) \sum_{q} \mathbf{Z}_{h,w,q} \log(\widehat{\mathbf{Z}}_{h,w,q})$
  - Z_hat represents the predictions and Z represents the ground truth values (dimensions of h,w,q)
  - The v function rebalances the classes to incentivize rarer colors
  - We sum over the the log of our predictions times the ground-truth for all the bins in a given height and width. This is then multiplied by the sample from the space calculated by v, and then we sum over all pixels in the image

Results

Due to Training Limitations (see issues section) We were only able to train on 100 Images with a bin distribution trained on 10 images.
- The predicted colors had to be scaled at this stage and our loss contained outliers

***Fig. 4**: Results demonstrate model didn't learn*

Issues

Problems:
- Calculating the nearest neighbors of a color for a soft-encoding was extremely slow
- We didn't implement K-means initialization
- Accuracy currently measures only how far our images are from the originals, not how plausible they look (the goal of the project

Solution Attempts:
- We've spent lots of time optimizing code, removing for loops, and using map_fn where we can.
- We have tried using @tf.function to make a graph out of our functions but this caused us significant issues with our implementation.
- We ended up training with a very small set of data to be able to get the results shown here

Future Work

With more time we would like to actually train our model and hopefully see it working
- This issue is discussed more above in the Issues section

Additional Metrics:
We'd like to implement more metrics for measuring the accuracy of our model:
- Added metric showing how often humans felt our output colorizations looked real
- Running a pre-trained object classifier on our output images and comparing with labels

Data:
- CIFAR-10 is a very small dataset limited to only 10 object types
- We'd like to train on a much larger dataset such as ImageNet

Object Classification:
- Our implementation focuses on finding plausible colors for objects where colors may vary
- In the future we'd like to investigate making a model that classifies objects as well in order to use that information to choose correct colors based on each object instead of just plausible

Social Media Fake News Detector

by Gradient Ascent: Ariel Rotter-Aboyoun (arottera), Daniel Kostovetsky (dkostove), Julius Sun (jsun6), Raymond Cao (rcao6)
CSCI 1470 (Deep Learning), Department of Computer Science, Brown University

Introduction

The proliferation of fake news on social media has become a well-publicized problem in recent years. Currently the biggest issue is the difficulty of verifying the authenticity and accuracy of content shared online. Since relatively few people take the time to extensively vet the news they share, developing an automated process to detect false content could significantly curb the spread of dangerous misinformation. We build a model that will classify public statements, such as social media posts, on the basis of whether they are real or fake, without prior knowledge of the subject domain. The model is based on the paper *Fake News Identification on Twitter with Hybrid CNN and RNN Models*, and, as the paper's title suggests, involves LSTMs and CNNs.

Figure 1. This fake social media post, posted by hackers who infiltrated the Associated Press account, went viral and caused a brief crash in the stock market.

Methodology

The model paper suggested three different architectures: a LSTM RNN, a LSTM with dropout regularization, and a LSTM with a 1D CNN. Few further details were given, other than a suggested dropout rate of 20%. We implemented all three architectures. We used a word embedding layer; then, either a dropout layer, a 1D convolution and pooling layer, or none of the above; then, an LSTM, and finally, a dense output layer with softmax activation. As suggested by the paper, we used a trial-and-error grid search to tune our hyperparameters, such as the learning rate, embedding size, and the size of the LSTM. General observation was also used to select the number of epochs for training.

Data

The dataset used in the model paper consisted of 5,800 tweets that reported on a news event, which were classified as either true or false. This dataset, however, is not publicly available. Instead, we used a similar dataset, known as LIAR, introduced in *"Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection*. LIAR consists of 13,000 public statements made by prominent political and media figures, including a validation set (10%) and test set (10%). They are classified into one of six truth categories (taken from the fact-checking website PolitiFact): true, mostly true, half true, barely true, false, or pants on fire. The dataset also contains additional information, such as the speaker of the statement and the occasion, although we ignored this information to maintain consistency with our model paper.

Statement	Truth Value
Over the past five years the federal government has paid out $601 million in retirement and disability benefits to deceased former federal employees.	True
Suzanne Bonamici supports a plan that will cut choice for Medicare Advantage seniors.	Half-True
In the case of a catastrophic event, the Atlanta-area offices of the Centers for Disease Control and Prevention will self-destruct.	Pants on Fire

Figure 2. Sample statements and truth labels from the LIAR dataset.

Results

Our best accuracy of 25.78% was achieved by the plain LSTM model with optimally tuned hyperparameters. All results are given below. We also assigned the truth classes numerical labels (0-5) and calculated the mean squared error for each model.

Model	Accuracy	Mean Squared Error
LSTM	25.78%	3.34296875
LSTM with Dropout	25.16%	3.3671875
LSTM with CNN	25.16%	3.3984375

Figure 3. Model result metrics.

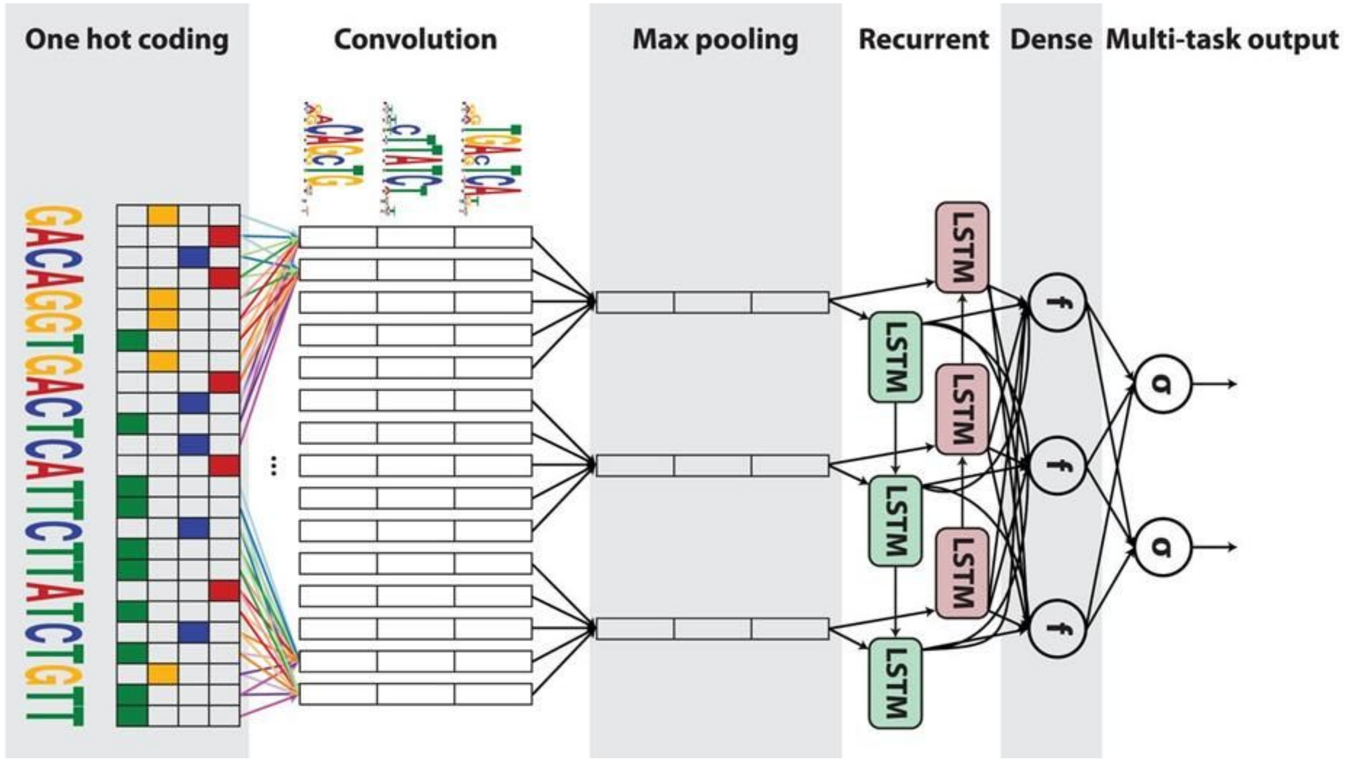
Statement	Prediction	Actual
My home state since June of 2009 created 40 percent of the new jobs in America.	mostly true	mostly true
Rebuilding three high schools will benefit 40 percent of Portland Public School students.	mostly true	pants on fire
Victory! Republicans by 2 to 1 vote to endorse Mark Neumann on first ballot at GOP convention.	half true	false
If you are a member of union, your median weekly income is roughly $200 more than if you are a nonunion member, and that doesn't include benefits.	half true	true

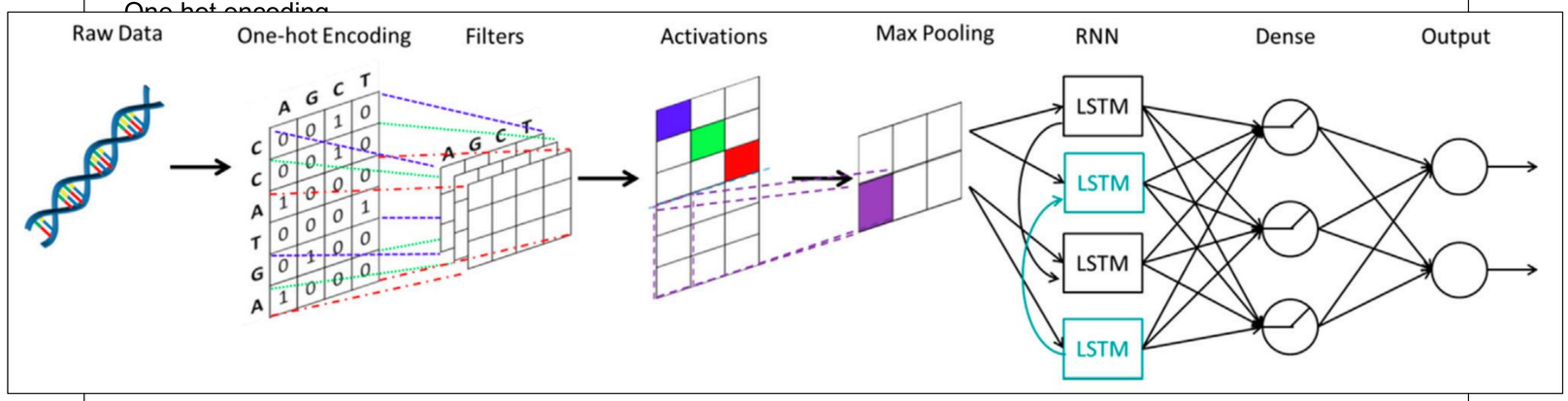
Figure 4. Sample correct and incorrect model predictions.

Discussion

Determining the truthfulness of a statement without context is a difficult problem. We were only able to achieve a 25.78% accuracy on our dataset, which may seem low, but is probably close to state of the art. For reference, we attempted to classify training examples by hand and achieved an accuracy that was far less than 25.78%, and in fact, no better than random guessing. Many statements in our dataset, such as "The economy bled $24 billion due to the government shutdown," are virtually impossible to factually evaluate without an external reference. Others, such as "We have a federal government that thinks they have the authority to regulate our toilet seats," are subjective, opinionated, or don't have a well-defined truth value. Some statements did contain certain words that made them identifiable as true, or more frequently, false. For instance, the sentence "In the case of a catastrophic event, the Atlanta-area offices of the Centers for Disease Control and Prevention will self-destruct," includes the word 'self-destruct' that is associated with fictional stories, implying that it is false. Perhaps our model learned to make classifications by looking for such words.

Future work could involve taking the additional context information in the LIAR dataset into account. Additionally, models may be given access to some factual database, such as pages of Wikipedia, for assistance. Perhaps a new dataset could be gathered where sentences are labeled with an 'unknown' truth value, giving models an escape hatch for opinionated or subjected inputs. It may also be useful to analyze the model during execution in order to more precisely determine what it looks for when making classifications.


One hot coding
Convolution
Max pooling
Recurrent
Dense
Multi-task output
GACAGGTGACTCATTCTTATCTGTT
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
f
f
f
σ
σ


Raw Data
One-hot Encoding
Filters
Activations
Max Pooling
RNN
Dense
Output
A G C T
C
C
A
T
G
A
LSTM
LSTM
LSTM
LSTM