Somit Gupta
somitg@Stanford.edu

# Annotating pathogenicity of genetic variants

A deep neural network solution

## Overview

**Motivation:** Genetic variations can help explain many diseases. Every human being has a unique genetic code and there are a lot of genetic variants within a group of individuals. Most of the deleterious genetic variants have been depleted from genome by natural selection. It is important to identify which genetics variations are likely to be pathogenic or deleterious. This will help researchers focus on the likely pathogenic genetic variants and accelerate the pace of diagnose & cure of many diseases.

**Background:** [1] Key challenges: i) There are a lots of annotations for alleles and the whole genome & other inputs with metrics that are rarely compatible with each other. ii) The data on current pathogenic mutations cannot be used for training as it suffers from a ascertainment bias. To overcome these challenges [1]**CADD** integrates diverse genome annotations and scoring any possible human single nucleotide variant (SNV) or small insertion/deletion (indel) event. The basis of CADD is a linear SVM model to contrast the annotations of fixed or nearly fixed derived alleles in humans relative to simulated variants. Deleterious variants – that is, variants that reduce organismal fitness – are depleted by natural selection in fixed but not simulated variation.

[2] **DANN** uses a Deep Neural Network model to **improve the accuracy from 58.2% to 66.1%.** Although DNN improves on the linear methods, its accuracy is still unsatisfactory. Potential factors: (i) training data has too much noise (ii) features currently used in genome annotation are insufficient (iii) model training needs further improvement

## My contribution

I have improved the accuracy of the classifier from **66.1% to 68.77%** by using a different DNN model.

**Data**: I am using the same data as DANN and CADD. The data referenced in DANN was already split into training, validation and testing set in 8:1:1 ratio. The data contains 29.4 million variants (half fixed or nearly fixed human derived alleles ("observed"), half simulated *de novo* mutations ("simulated")) and 63 distinct annotations. Due to the coding of categorical values using Boolean variables, the total number of features in this model is 949.

**Model**: While DANN used 3 hidden layers with 1000 units, 10% dropout and tanh activation to minimize the cross entropy loss using batch stochastic gradient descent with a momentum rate that increases from 0.01 to 0.99 linearly for the first 10 epochs and then remains at 0.99.

My final model uses 6 hidden layers with 3000 units, 2000 units in the first two layers, and 1000 units in the remaining layers. Each with 20% dropout and relu activation to minimize the cross entropy loss using Nadam stochastic gradient descent algorithm.

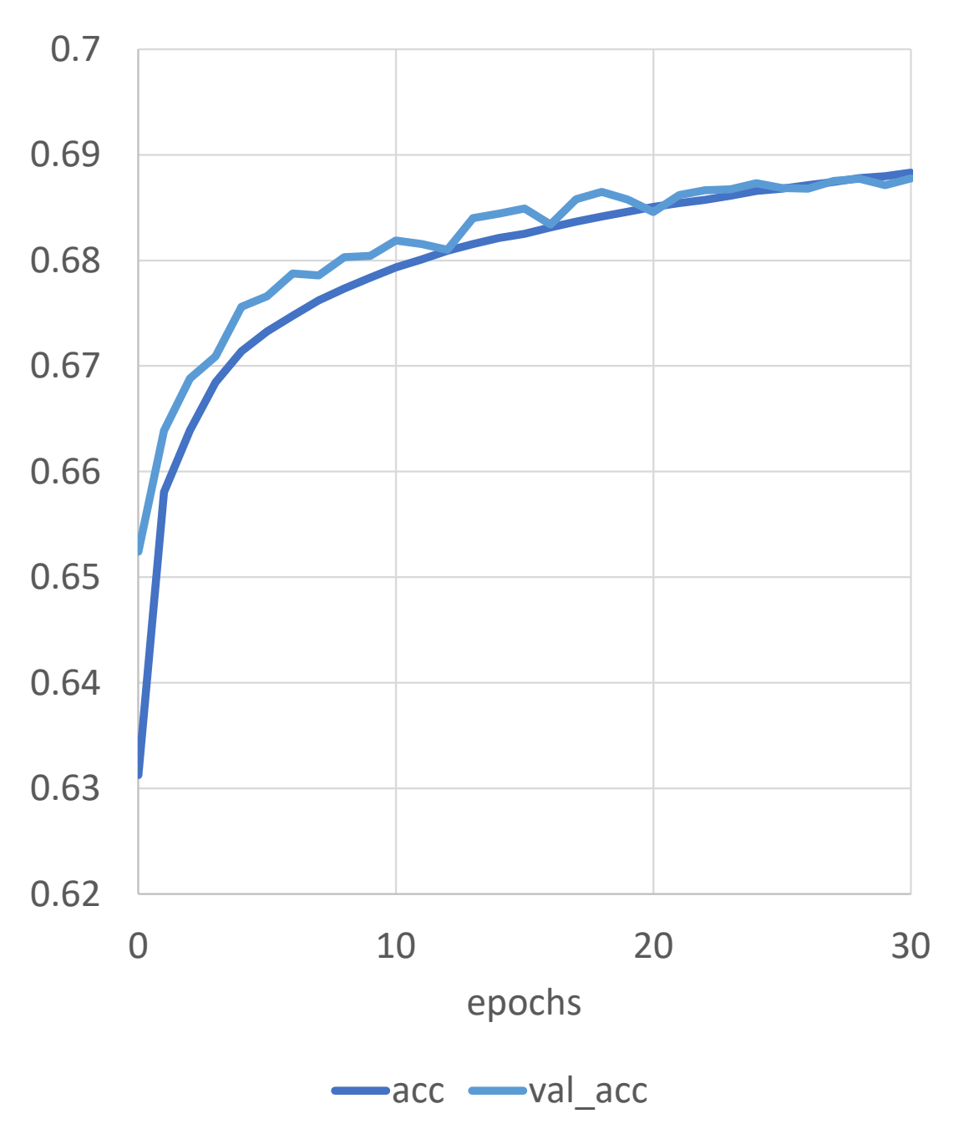

**Fig1:** Training and validation set accuracy

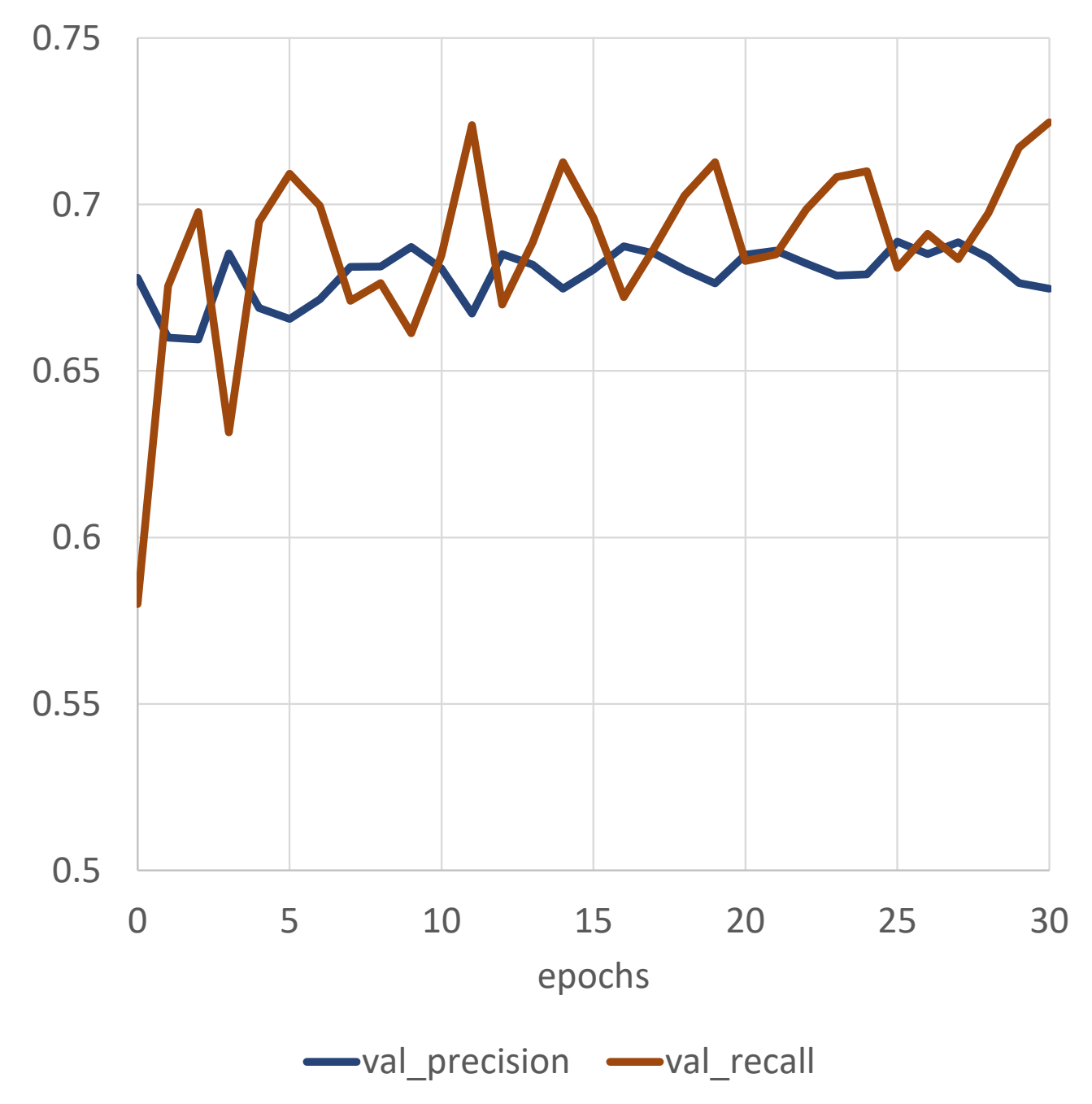

**Fig2:** Validation set precision and recall. Recall seems to be more volatile than precision over different epochs

## Developing Deep learning models on large data sets

Problem: Developing a deep learning model is an iterative task. You need to explore multiple options for number of hidden layer, units, learning rate, momentum, batch size. The data size is large (26M x 1000). It costs time and money to train a single model. For 20 epochs, it can take around 10 hours and 20$ (AWS EC2 compute cost) – varies from model to model. This makes it very costly in terms of time and money to even tune parameters based on grid search.

I used a parts of the following strategy to address the problem (the description below also includes some things I realized in hindsight).

**Phase 1: Use a small (10%) random sample of the training set**
Objective: Reduce the training error quickly to have less bias in the model
Metric: Training error after 20 epochs

1. Start with an initial model.
2. Chose the best among models with different activation functions and optimization functions.
3. Choose the best among two models with same number of units - Model 1: Double the units in the hidden layers & tune parameters, Model 2: double the number of hidden layers in the model & tune parameters.
4. Repeat 3 until you get low training error.
5. To reduce the divergence between training and validation error. Trim the model down – reduce the number of units in the later hidden layers, or reduce the number of layers.

**Phase 2: Use the full training set**
Objective: Reduce the validation error
Metric: Validation error after 20 epochs

1. Start with the model selected in phase 1
2. If the variance is high, add regularization parameters like dropout
3. Further tune the learning rate, momentum and other parameters of the model to allow the model to converge faster
4. Run the final model for higher number of epochs to get a model with least validation error

**MINIMIZING THE TRAINING LOSS ON 10% SAMPLE DATA SET**

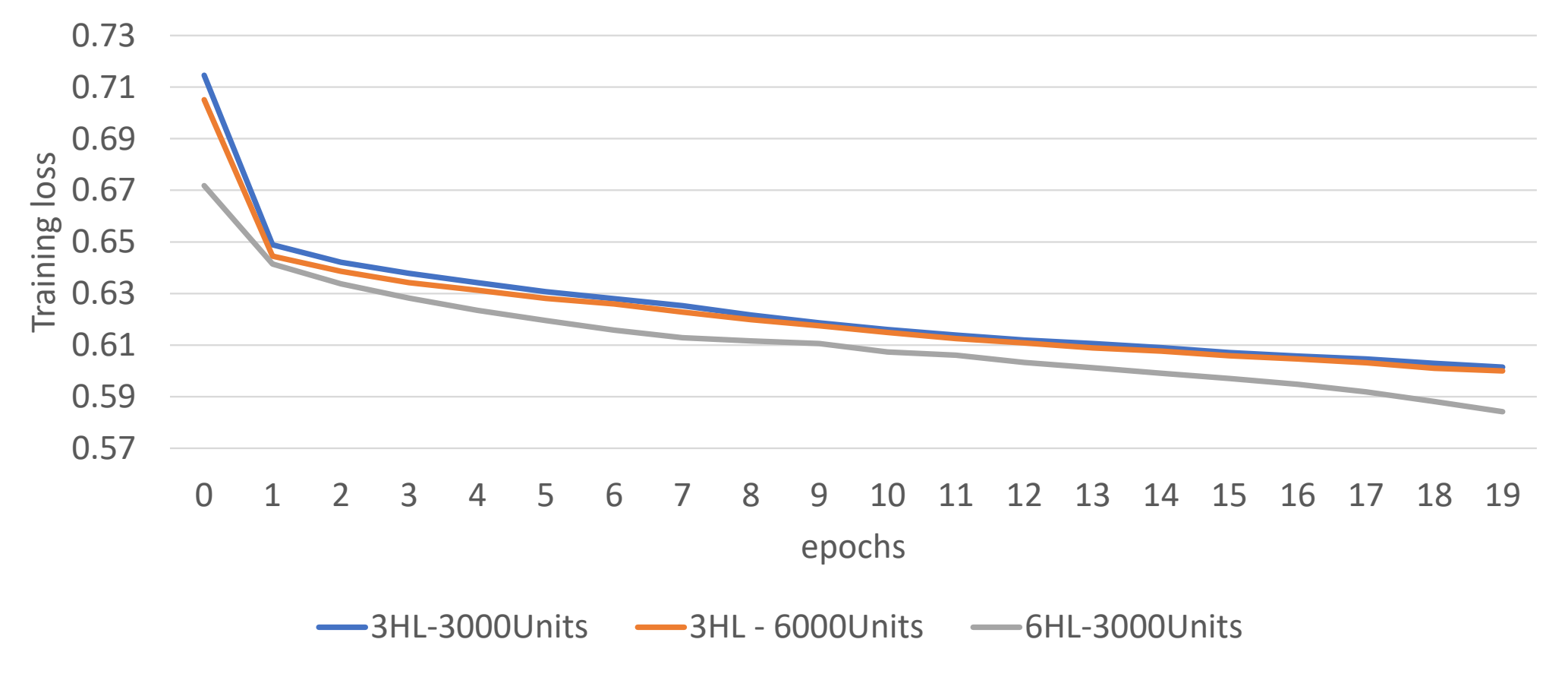

**Fig3:** For the same number of hidden layer, a DNN with 6 hidden layers and 3000 units each better minimizes the training loss than a DNN with 3 hidden layers and 6000 units each.

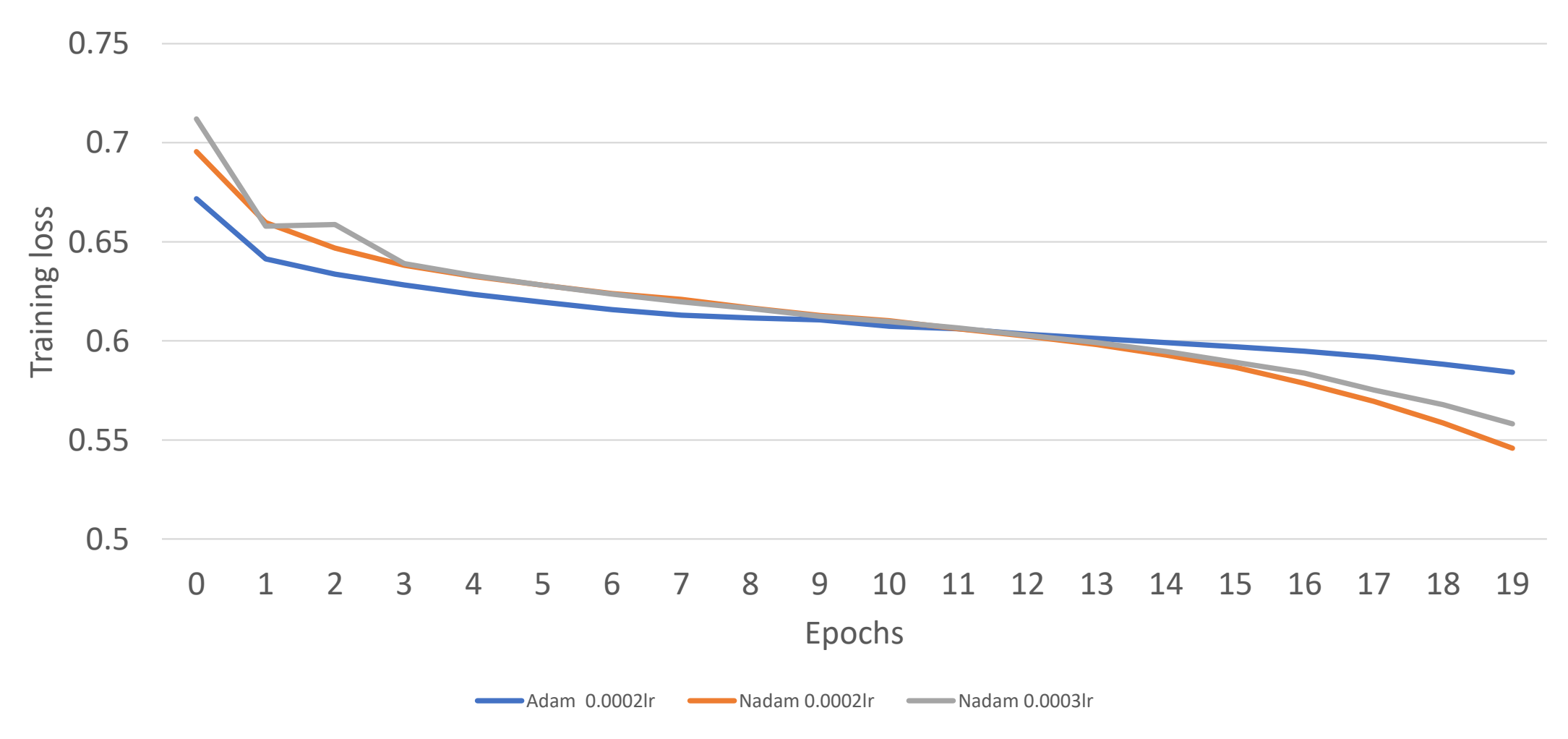

**Fig4:** For the network with 6 hidden layers and 3000 units each, Nadam optimizer with a learning rate of 0.0002 performs better than other tired options.

**TRIMMNG: MINIMIZING THE DIVEREGENCE IN VALIDATION AND TRAINING ERROR ON THE 10% SAMPLE DATA SET**

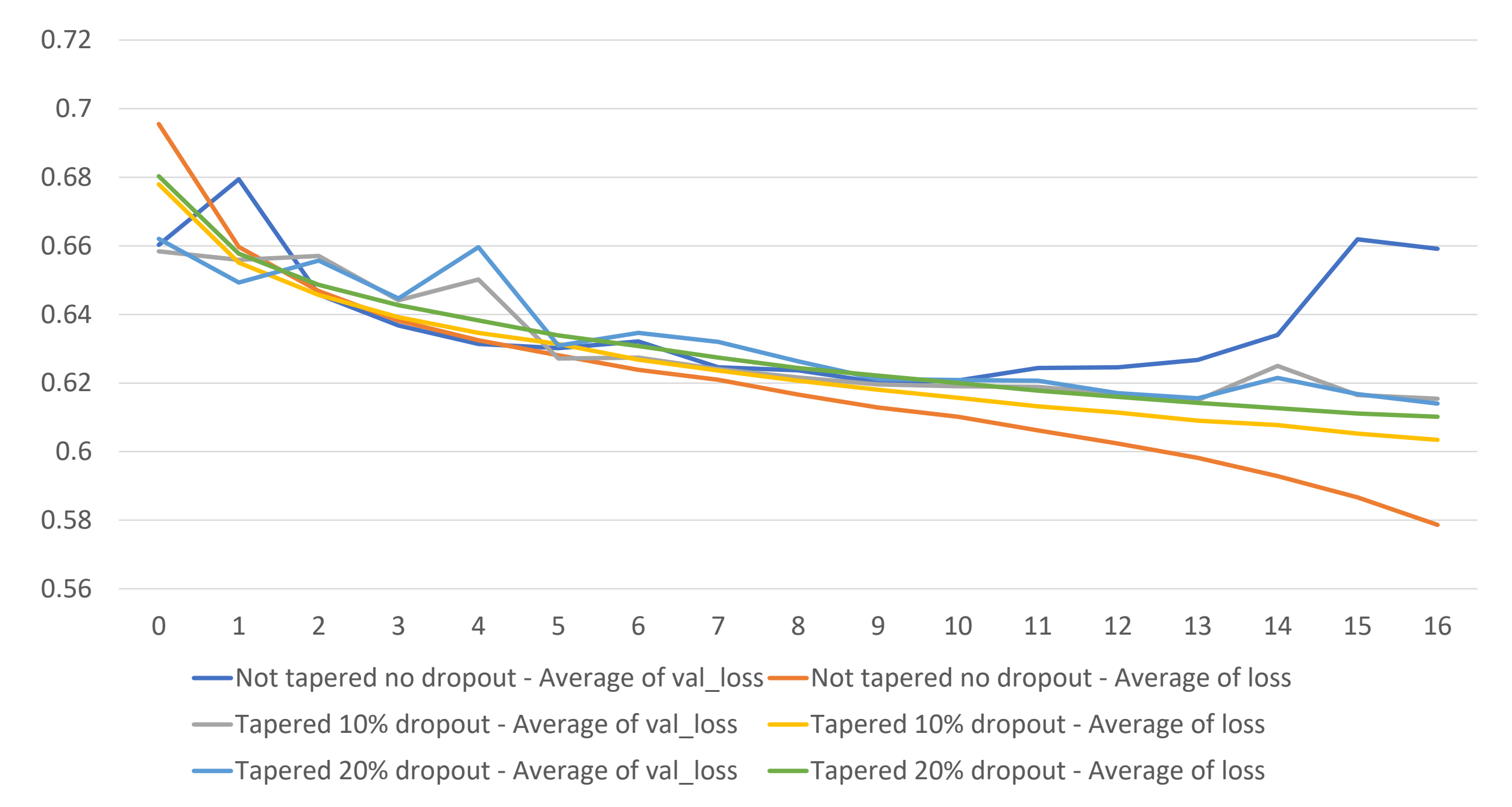

**Fig5:** Trimming. A tapered model with 6 hidden layers, 3000 units in the first hidden layer, 2000 units in the second hidden layer, and 1000 units in the rest of the hidden layers with 20% dropout minimizes the divergence between validation and training error to avoid overfitting. The overfitting will be further reduced when we train the model on full training set.

## Future improvements

1. Improve the model training with better tuning of parameters. The model reaches around 68% validation accuracy quite fast. Moving beyond that has been much slower
2. Analyze the model on known pathogenic variants. As explained below current model has been trained on a synthetic data set. We need to understand how well this model would do on real data.

## References

1. Kircher M., et al. (2014) A general framework for estimating the relative pathogenicity of human genetic variants. Nat. Genet., 46, 310–315.
2. Quang D., et al. (2015) DANN: a deep learning approach for annotating the pathogenicity of genetic variants. Bioinformatics,
3. http://cs231n.github.io/neural-networks-3/
4. http://sebastianruder.com/optimizing-gradient-descent/
5. Nuts and Bolts of Applying Deep Learning (Andrew Ng) https://www.youtube.com/watch?v=F1ka6a13S9I
6. Nadam: http://cs229.stanford.edu/proj2015/054_report.pdf
7. http://www.deeplearningbook.org/contents/optimization.html#pf10