2nd Place Solution - with Single 1D-CNN (Private LB: 0.01601)

Mechanisms of Action (MoA) Prediction
Baosen Guo (Team: tmp)
Email: baosenguo@163.com
Private LB Score: 0.01599 (2nd place)

1. Overview

First of all, many thanks to Kaggle team and Laboratory for Innovation Science at Harvard for providing priceless dataset and hosting this great challenge. This is the my first kaggle competition, I have to say I learned a lot from the valuable community.

This approach consists of 3 single modes:

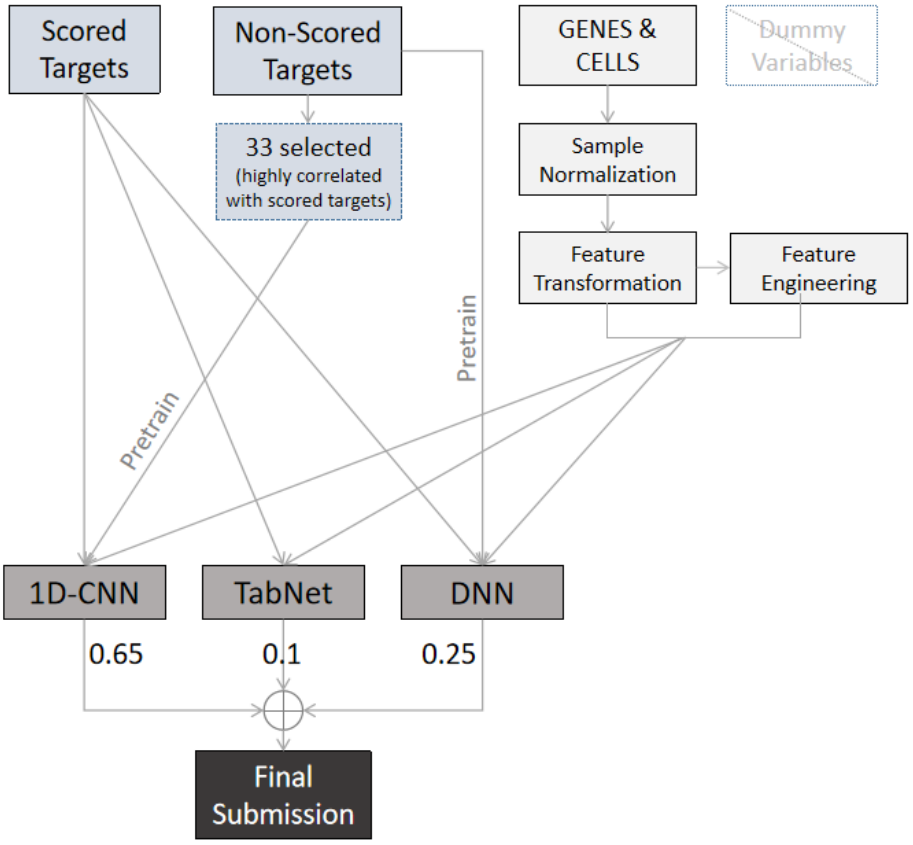
- 1D-CNN
- TabNet
- DNN

and the final submission is generated by weighted average of single models outputs.

The table below lists the performance of the single modes and the final blending in this scheme. The most important part of this scheme may be 1D-CNN **0.01601** (private lb). The final blending only improves by 0.00002 on this basis (private lb 0.01599).

		Local CV	Public LB	Private LB
Single Model	1D-CNN	0.01672	0.01806	**0.01601**
	Tabnet	0.01684	0.01835	0.01621
	DNN	0.01690	Not submitted	Not submitted
Blend	1D-CNN * 0.65 + Tabnet * 0.1 + DNN * 0.25	0.01666	0.01804	0.01599

The flow chart below outlines the technical details , I will introduce it in the following sections.



2. Local CV

One of the most important objectives of this challenge is to test the performance of the model for drugs that have never appeared in the public dataset.

For this, I used the cv stratification code posted by @cdeotte (https://www.kaggle.com/c/lish-moa/discussion/195195).

Generally in genomics and biomedical data, sample type, instrument error, reagent quality and other systematic deviation sometimes affect the distribution significantly. This is inevitable in the real world, so proper assessment is very important. Here's an example: when batch effect exists, "leave batch out" can reflect the generalization ability of the model more objectively than random kfold-split.

In this competation, although not very sure, this "leave drug out" strategy seems to reduce the risk of over fitting in a specific group of drugs (public lb 4th; private lb 2nd).

3. Pre-processing

Sample Normalization

There are some assumptions (may be wrong):

- The raw biological data are often not comparable due to the inevitable existence of systematic errors(e.g., data cannot be produced on the same machine, experiments cannot be performed by the same person)

- Even with totally different treatments, the number of significantly different features won`t be too large.
- It can be simply assume that the distributions of most samples are similar.

Based on these assumptions, samples weres normalized by the distribution of gene expression and cell viability separately.

In short, the gene data in each sample is subtracted from the mean value of 25% and 75% quantiles .
Similarly, the cell data is subtracted from the mean of 25% and 72% quantiles, and then divided by the 4 + 75% quantiles of the new distribution. The slightly different steps for genes and cells are determined by the normality of the distribution after treatment.
Generally, samples normalization should be implemented before features conversion.

Feature Transformation
After sample normalization, quantile transformation is applied on numerical features.

4. Feature Engineering

There is nothing special in feature engineering, even no variance filtering in 1D-CNN model. Different feature processing methods in the other two models are only used to increase the diversity.

- PCA feature are used for all models with different n_components (50 genes + 15 cells for 1D-CNN, 600 genes + 50 cells for TabNet and DNN).
- Statistical features (such as sum, mean) and combination features are used in Tabnet,
- Variance fillter is used in Tabnet and DNN.
- In addition, dummy variable (cp_time, cp_dose) are removed in all models.

5. Loss

To deal with the imbalance of targets and reduce overfitting to specific ones, BECloss with label smooth and pos_weight was used in all single models.
As mentioned by others, label smoothing performed well by reducing the overconfident probability .
In addition, targets weight is slightly adjusted by the reciprocal of class frequency.
Specifically, for each target i, the weight is set to log (Fmin + 100) / log (Fi + 100), where the Fi is the number of positive samples in target i, and Fmin donates the min positive count of all target. The constant term 100 is added to prevent the target with low frequency (e.g., with 1 positive sample) from affecting model extremely, and log conversion is to make it smooth.

6. Modelling

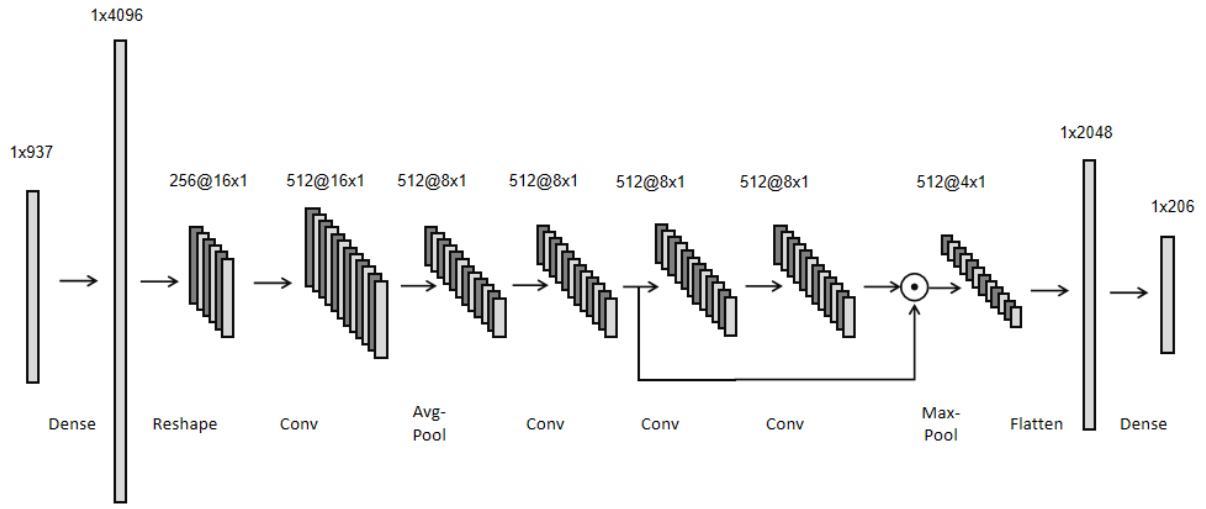
1) 1D-CNN

This single mode achieves the best performance in this approach (private score : **0.01601**).
Using such a structure in tabular data is based on the idea that:

- CNN structure performs well in feature extraction, but it is rarely used in tabular data because the correct features ordering is unknown.
- A simple idea is to reshape the data directly into a multi-channel image format, and the correct sorting can be learned by using FC layer through back propagation.

Model Architecture

Based on these ideas, the model which extracts features through 1D-CNN (performs better than 2D and 3D in experiments) are implemented. The figure below shows the main structure.



As shown above, feature dimension is increased through a FC layer firstly. The role of this layer includes providing enough pixels for the image by increasing the dimension, and making the generated image meaningful by features sorting.
Next, data is directly reshaped into image format (size 16*1, chanel 256).
Like the basic CNN model, features are extracted in the next several 1D-Conv layers with a shortcut-like connection.
Finally, the extracted features are used to predict targets through a FC layer after flatten.

Pre-training

Additionally, non-scored-targets are used alone for pre-training. In this step, only 33 non-scored-targets that are most correlated to the scored-target are trained for only 1 epoch. The purpose of this is only to make the model better initialised. Irrelevant targets and more training steps may lead to unexpected consequences.

2) TabNet

TabNet used in this approach is slightly modified from a public kernel (https://www.kaggle.com/kushal1506/moa-tabnet-inference) . There is nothing special except the

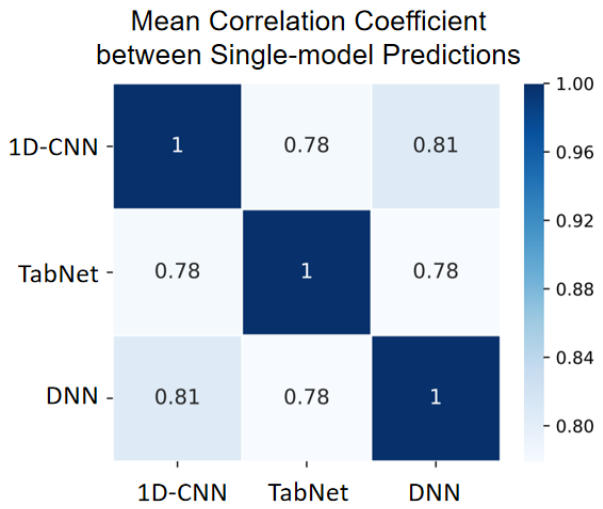
preprocessing method mentioned above.

3) DNN

Similarly, the DNN model is a modified version learned form public notebook (https://www.kaggle.com/thehemen/pytorch-transfer-learning-with-k-folds-by-drug-ids/comments), with pre-training on all targets (scored & non-scored) and fine-tuning on scored ones.

7. Blending

3 single models are blended as the final submission. Compared with 1D-CNN, the final blending only improves by 0.00002 (private lb 0.01599), perhaps due to the lack of diversity or the insufficient number of single modes.



8. Source Code

You can find training and inference code here: https://github.com/baosenguo/Kaggle-MoA-2nd-Place-Solution