Iowa Initiative for Artificial Intelligence
Final Report

Project title:	Machine Learning Classification of Two-Dimensional Infrared Spectra for High Throughput Screening
Principal Investigator:	Evan Schroeder and Christopher M. Cheatum
Prepared by (IIAI):	Avinash Mudireddy
Other investigators:	
Date:	02/25/2023
Were specific aims fulfilled:	Partially
Readiness for extramural proposal?	No
If yes … Planned submission date	N/A
Funding agency	N/A
Grant mechanism	N/A
If no … Why not? What went wrong?	Aim 1 is complete, Aim 2 partially complete. The obtained results for 5 values of SNR are insufficient for a subsequent proposal. (Aim 2 needs to be finished)

Brief summary of accomplished results:

Research report:
Aims (provided by PI):

We propose to use machine learning to classify measurements of enzyme dynamics for high-throughput screening.
Our group specializes in a kind of spectroscopy called two-dimensional infrared (2D IR) spectroscopy that measures protein dynamics on the femtosecond to picosecond timescale. We have previously shown that this method is sensitive to functional motions in enzymes, and we currently have a funded NIH R21 project to develop 2D IR for high-throughput screening approaches to drug discovery. Our current methods of data analysis, however, require many measurements to determine the dynamics and thereby identify changes resulting from effector binding. We propose to use machine learning to classify spectra as a way of identifying potential hits with fewer measurements.

The central objective of this proposal is to evaluate the appropriateness and performance of machine learning as a tool for the analysis of 2D IR spectra. While previous work from our group has shown the ability of machine learning methods to classify 2D IR spectra exhibiting large differences, the relative differences of various enzyme-effector pairs are generally unknown and potentially small. Therefore, we propose to test the limits of machine learning classification of 2D IR spectra. If successful, this approach will establish the feasibility of machine learning for use in high-throughput screening using 2D IR and will produce the necessary proof-of-concept results to motivate continued development of machine learning augmented 2D IR for screening allosteric effectors in drug discovery. To achieve the goals of this project we propose the following aims:

Aim 1. Classify 2D IR spectra via machine learning methods. Classification of allosteric effectors will need to discriminate nuanced and potentially small differences. As a proof-of-concept, we propose using machine learning methods to classify simulated 2D IR spectra into the categories of "Same" or "Different". Simulated spectra have the advantage of providing an essentially unlimited dataset as well as using input parameters set by the user such that the true value is known *a priori.* We have existing experimental data of modified enzymes whose spectra exhibit quantifiable differences. Comparing the magnitude of these differences with the magnitude of differences detectable by machines learning will establish the efficacy of machine learning augmented 2D IR for screening allosteric interactions.

Aim 2. Determine the minimum number of data points needed for classification. Our current method of analysis requires many high-resolution spectra. However, based on preliminary work in our group, we believe accurate classification can be accomplished with a significantly reduced dataset. We aim to utilize machine learning to not only reduce the number of data points need for classification, but to also identify the minimum dataset size for accurate classification. These studies will enable the application of 2D IR in a rapid screening protocol.

Data for Aims:
Each two-dimensional spectrum is composed of a complex valued intensity as a function of pump-time (x-axis) and probe-frequency (y-axis), respectively. The frequency axis spans an ~150-pixel range. The time axis range is typically ~167 time points for our experiments. The slices of intensity vs time yield a free induction decay (FID) for each frequency. Thus, each two-dimensional spectrum consists of 150 separate FIDs.

The two-dimensional spectra are then collected as a function of a second time domain T_W, which typically includes anywhere from a few to a hundred distinct 2D IR spectra each at a different value of T_W. Thus, in total, the 2D IR data form a complex-valued three-dimensional data cube as a function of two time axes and a frequency axis. For both time axes, the time intervals and number of points can be varied independently.

The simulated 2D IR spectrum for a single sample is generated by defining two key line-shape parameters: Δ and τ. To generate a dataset of samples, a reference $(\Delta_{ref}, \tau_{ref})$ is selected and samples are generated by varying the (Δ_i, τ_i) parameters of sample index i. The dataset is divided into two categories: “Same” and “Different”. If the percent difference magnitude of the sample (Δ_s, τ_s) relative to the reference is larger than 10%, the sample is categorized as “Different”; otherwise the sample is categorized as “Same”. Each category is composed of 1000 samples. Additional datasets are generated for different values of signal-to-noise (SNR) by adding gaussian noise to the simulated spectra. The 2D IR spectrum for each sample is normalized such that the maximum value does not fall outside the range $[(-1-1i), (1+1i)]$, while maintaining the scaling between each two-dimensional spectrum.

AI/ML Approach:

In this report, we explore the use of student-teacher distiller networks for the purpose of classification. The goal of our research is to improve the accuracy of classification models by leveraging the knowledge of larger, more complex models (i.e., the teacher) to train smaller, more lightweight models (i.e., the student). The distillation process involves transferring the knowledge learned by the teacher model to the student model through a combination of mimicking the teacher's predictions and learning from its mistakes. Our approach is particularly relevant for scenarios where the computational resources available for training and inference are limited, making it challenging to deploy large models in real-world applications. By distilling the knowledge from larger models into smaller ones, we aim to achieve a good balance between accuracy and efficiency, making our models suitable for deployment in resource-constrained environments.

Data Pre-processing:

For this classification problem, we perform several data preprocessing steps. Firstly, we create two folders, one for "SAME" and the other for "DIFF", each containing a set of 1000 sample images. We randomly shuffle the final paths in the folders and split the images into training and testing sets.

For each image, read both the real and imaginary components of the image data. We then stack the real and imaginary components of each image together to form the teacher data. However, for the student data we reduce the size of the input data by picking every tenth value in both the pump-time and T_w axes. Hence the Student data is 1/100th the size of teacher data. Finally, we normalize the data to ensure that the data values fall within a specific range. The resulting preprocessed data includes the teacher data and student data for both the training and testing sets, as well as the corresponding numerical labels for DIFF and SAME categories for each set.

Model design:
Both teacher and student networks are based on the MobileNet architecture, which is a popular convolutional neural network architecture for efficient and mobile-friendly image classification.

The MobileNet blocks consist of depthwise separable convolution layers. A depthwise separable convolution layer consists of two parts: a depthwise convolution and a pointwise convolution.

The depthwise convolution applies a separate convolutional filter to each input channel, which allows the model to capture spatial information while reducing the number of parameters. This is followed by a pointwise convolution that applies a 1x1 convolutional filter to combine the outputs of the depthwise convolution. The pointwise convolution allows the model to learn complex, non-linear representations of the input data.

In both the teacher and student networks, there are multiple MobileNet blocks stacked on top of each other. These blocks are separated by a down sampling layer, which reduces the spatial dimensions of the feature maps while increasing the number of channels.

The overall structure is:
Conv block
MobileNet block
Conv Block
Dense Layers
Output Layers.

However, The Distiller class takes the following parameters during initialization:

- teacher: the pre-trained teacher network
- student: the student network to be trained
- temperature: the temperature parameter used for the softmax function during knowledge distillation
- alpha: the weight assigned to the KD loss during training

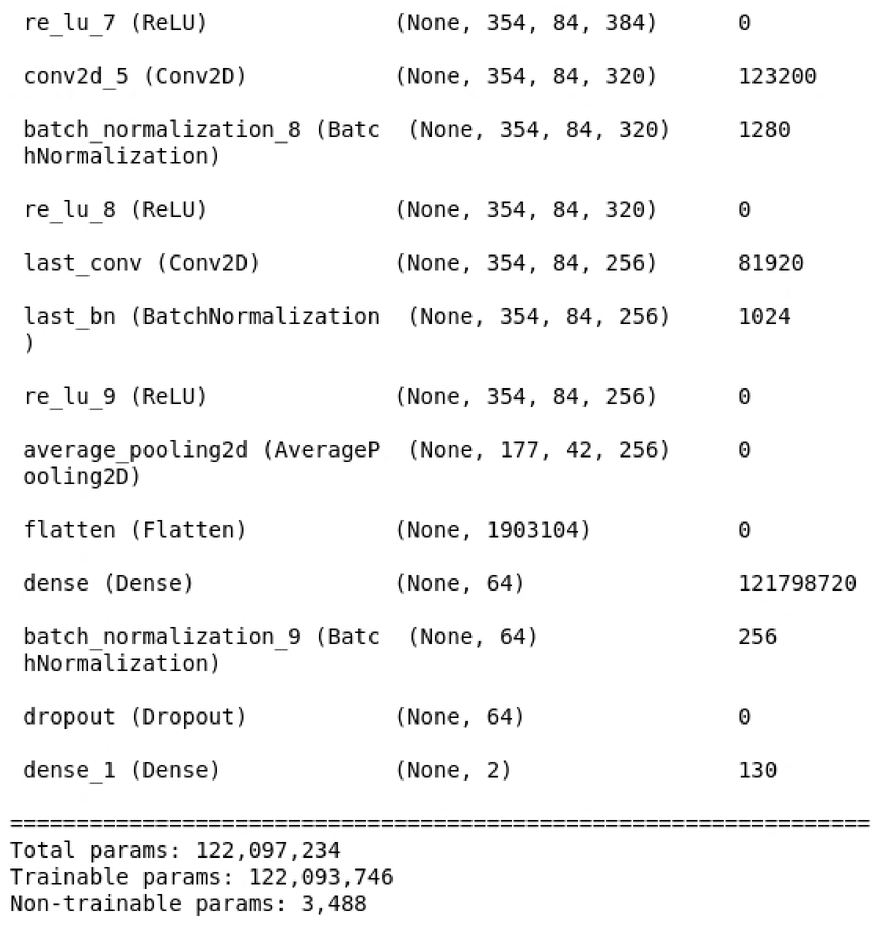
The Distiller class has a train method, which performs the training of the student network using knowledge distillation. During each training epoch, the method iterates over the training data and computes the KD loss between the teacher and student outputs. The overall loss for the student network is a weighted sum of the KD loss and the original cross-entropy loss between the student network predictions and ground truth labels.

Experimental methods, validation approach:

As a part of the experimental setup, for each of the SNR categories,
The train: test split is 80:20.

Teacher Network:

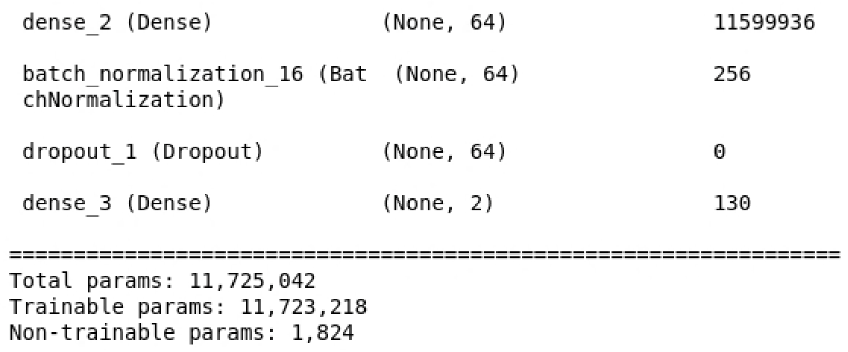
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_1 (InputLayer)        [(None, 708, 167, 166)]   0         
                                                                 
 conv2d (Conv2D)             (None, 708, 167, 32)      47840     
                                                                 
 batch_normalization (BatchN  (None, 708, 167, 32)     128       
 ormalization)                                                   
                                                                 
 re_lu (ReLU)                (None, 708, 167, 32)      0         
                                                                 
 depthwise_conv2d (Depthwise  (None, 708, 167, 32)     320       
 Conv2D)                                                         
                                                                 
 batch_normalization_1 (Batc  (None, 708, 167, 32)     128       
 hNormalization)                                                 
                                                                 
 re_lu_1 (ReLU)              (None, 708, 167, 32)      0         
                                                                 
 conv2d_1 (Conv2D)           (None, 708, 167, 16)      528       
                                                                 
 batch_normalization_2 (Batc  (None, 708, 167, 16)     64        
 hNormalization)                                                 
                                                                 
 re_lu_2 (ReLU)              (None, 708, 167, 16)      0         
                                                                 
 conv2d_2 (Conv2D)           (None, 708, 167, 96)      1632      
batch_normalization_3 (Batc  (None, 708, 167, 96)     384       
hNormalization)                                                 
                                                                
re_lu_3 (ReLU)              (None, 708, 167, 96)      0         
                                                                
depthwise_conv2d_1 (Depthwi  (None, 354, 84, 96)      960       
seConv2D)                                                       
                                                                
batch_normalization_4 (Batc  (None, 354, 84, 96)      384       
hNormalization)                                                 
                                                                
re_lu_4 (ReLU)              (None, 354, 84, 96)       0         
                                                                
conv2d_3 (Conv2D)           (None, 354, 84, 64)       6208      
                                                                
batch_normalization_5 (Batc  (None, 354, 84, 64)      256       
hNormalization)                                                 
                                                                
re_lu_5 (ReLU)              (None, 354, 84, 64)       0         
                                                                
conv2d_4 (Conv2D)           (None, 354, 84, 384)      24960     
                                                                
batch_normalization_6 (Batc  (None, 354, 84, 384)     1536      
hNormalization)                                                 
                                                                
re_lu_6 (ReLU)              (None, 354, 84, 384)      0         
                                                                
depthwise_conv2d_2 (Depthwi  (None, 354, 84, 384)     3840      
seConv2D)                                                       
                                                                
batch_normalization_7 (Batc  (None, 354, 84, 384)     1536      
hNormalization)                                                 
```



```
 re_lu_7 (ReLU)              (None, 354, 84, 384)      0

 conv2d_5 (Conv2D)           (None, 354, 84, 320)      123200

 batch_normalization_8 (Batc  (None, 354, 84, 320)     1280
 hNormalization)

 re_lu_8 (ReLU)              (None, 354, 84, 320)      0

 last_conv (Conv2D)          (None, 354, 84, 256)      81920

 last_bn (BatchNormalization  (None, 354, 84, 256)     1024
 )

 re_lu_9 (ReLU)              (None, 354, 84, 256)      0

 average_pooling2d (AverageP  (None, 177, 42, 256)     0
 ooling2D)

 flatten (Flatten)           (None, 1903104)           0

 dense (Dense)               (None, 64)                121798720

 batch_normalization_9 (Batc  (None, 64)               256
 hNormalization)

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 2)                 130

=================================================================
Total params: 122,097,234
Trainable params: 122,093,746
Non-trainable params: 3,488
```

Student network:

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_2 (InputLayer)        [(None, 708, 17, 17)]     0         
                                                                 
 conv2d_6 (Conv2D)           (None, 354, 9, 32)        4928      
                                                                 
 batch_normalization_10 (Bat  (None, 354, 9, 32)       128       
 chNormalization)                                                
                                                                 
 re_lu_10 (ReLU)             (None, 354, 9, 32)        0         
                                                                 
 depthwise_conv2d_3 (Depthwi  (None, 354, 9, 32)       320       
 seConv2D)                                                       
                                                                 
 batch_normalization_11 (Bat  (None, 354, 9, 32)       128       
 chNormalization)                                                
                                                                 
 re_lu_11 (ReLU)             (None, 354, 9, 32)        0         
                                                                 
 conv2d_7 (Conv2D)           (None, 354, 9, 16)        528       
                                                                 
 batch_normalization_12 (Bat  (None, 354, 9, 16)       64        
 chNormalization)                                                
                                                                 
 re_lu_12 (ReLU)             (None, 354, 9, 16)        0         
                                                                 
 conv2d_8 (Conv2D)           (None, 354, 9, 96)        1632      
                                                                 
 batch_normalization_13 (Bat  (None, 354, 9, 96)       384       
 chNormalization)                                                
                                                                 

 re_lu_13 (ReLU)             (None, 354, 9, 96)        0         
                                                                 
 depthwise_conv2d_4 (Depthwi  (None, 354, 9, 96)       960       
 seConv2D)                                                       
                                                                 
 batch_normalization_14 (Bat  (None, 354, 9, 96)       384       
 chNormalization)                                                
                                                                 
 re_lu_14 (ReLU)             (None, 354, 9, 96)        0         
                                                                 
 conv2d_9 (Conv2D)           (None, 354, 9, 320)       31040     
                                                                 
 batch_normalization_15 (Bat  (None, 354, 9, 320)      1280      
 chNormalization)                                                
                                                                 
 re_lu_15 (ReLU)             (None, 354, 9, 320)       0         
                                                                 
 last_conv (Conv2D)          (None, 354, 9, 256)       81920     
                                                                 
 last_bn (BatchNormalization  (None, 354, 9, 256)      1024      
 )                                                               
                                                                 
 last_relu (ReLU)            (None, 354, 9, 256)       0         
                                                                 
 re_lu_16 (ReLU)             (None, 354, 9, 256)       0         
                                                                 
 average_pooling2d_1 (Averag  (None, 177, 4, 256)      0         
 ePooling2D)                                                     
                                                                 
 flatten_1 (Flatten)         (None, 181248)            0         
```



```
 dense_2 (Dense)             (None, 64)                11599936

 batch_normalization_16 (Bat  (None, 64)               256
 chNormalization)

 dropout_1 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 2)                 130

=================================================================
Total params: 11,725,042
Trainable params: 11,723,218
Non-trainable params: 1,824
```

We used Adam with following configuration:
optimizer1 = tf.keras.optimizers.Adam(
  learning_rate=1e-6, beta_1=0.9, beta_2=0.999, epsilon=1e-07, amsgrad=False,
  name='Adam', decay= 1e-5, clipvalue=0.5)
categorical_crossentropy loss for both teacher and student network. KLDivergence loss for Distiller network.

All the models are trained for 100 epochs with 32 as batch size.

Results:

The results for the model are as follows.

	Teacher Network AUC			Student Network AUC		
Experiment name	Training AUC	Training Accuracy	Test Accuracy	Training AUC	Training Accuracy	Test Accuracy
Dataset 5- 1000 examples - SNR 100	99.6	98.12	95.75	99.2	97.19	96.5
Dataset 5- 1000 examples - SNR 50	98.64	95.19	95.5	99.13	96.69	97.25
Dataset 5- 1000 examples - SNR 5	97.03	93.06	90.75	95.54	89.88	90.5
Dataset 5- 1000 examples - SNR 2	92.97	86.81	86.25	89.74	83.44	83.25
Dataset 5- 1000 examples - SNR 0	51.39	50.56	46.25	48.21	47.87	45.25

Previous work by the Cheatum group has shown the following: that categorization of 2D IR spectra via artificial neural networks is possible (trained a network to categorize simulated spectra), and that categorization is still possible even with low SNR (trained a network to categorize experimental spectra at various SNR). The simulated spectra were able to be accurately classified (~99%) even when only one real-valued spectrum at a single T_w was selected, corresponding to a data-reduction of 334x. The simulated spectra were complicated by generating spectra with two pairs of (Δ, τ) instead of one, which resulted in ~94% accuracy using 3 T_w values (111x data-reduction). The experimental spectra were able to be accurately classified (100% at 100 SNR) for three real-valued spectra, corresponding to a data-reduction of 111x. However, this previous work was only validated for samples with relatively large spectral differences between categories, which the present study addresses.

In this study, sample spectra were simulated such that the differences between their simulation parameters were relatively small. It was shown that despite the small spectral differences, accurate categorization was still possible, even when using a data-reduction of 100x. Thus, aim 1 is considered completed.

Due to the time-span of this six-month study, it was infeasible to study the minimum number of data points needed for classification. This would have required generating a plot of “accuracy vs. data-reduction” for various SNR datasets. Thus, aim 2 is considered incomplete.

Ideas/aims for future extramural project:

It should be noted that the SNR 0 dataset should result in 50% accuracy, as the entire dataset is only gaussian noise. Thus, test accuracies for this study should be taken with ~5% uncertainty. Further work could investigate if this discrepancy is due to bias in the test set, or due to bias in the network.

An additional objective that was discussed was to complicate the spectra by adding a second (Δ, τ) pair when generating samples, as was done in the previous Cheatum group simulation studies. This was not attempted due to time constraints.

An idea was proposed that, instead of manually picking which data-points were most important for classification, to train a header network to pick the points automatically that would then be fed to the classification network. This was not attempted due to time constraints.