Ensemble Automator - Final Report

Connor (ctowler3) , Qasim (qasim), Joey (josehpdyer)

Abstract

Our goal was to make ML models that can express uncertainty and explainability in addition to making predictions. Deep Learning is known to operate under a black box. Quantifying the uncertainty in our predictions and explaining what combination of weights led to what prediction is an important part in explainability.

1 Actor-Critic Network for explainability

We thought an Actor-Critic network (ACN) would be helpful in showing the various features impacts on the prediction or label. The architecture of this network is similar to how ACNs are commonly made. The initial thought was the actor will be feed inputs and their predictions and try to explain them, the critic would then would then evaluate ('grade') how accurate those explanations were. The goal of the network would be to maximize the grade or reward the critic when evaluating the explanation, iteratively improve through feedback loops.

1.1 What is Truth

It was hard to measure truth in the sense, how do we grade how accurate the machines explanation of the prediction is. Truth is subjective by nature.

1.2 Our Implementation

Our initial state is an array length n of all 1s [1 1 1 1 ... 1] n times where n is the number of features in our Dataset. we map those values of the initial state array to the exponent of an x like the equation (1.1) where there is an x value for each feature (total n x values). We allow the exponents of numbers in the state array to range between -10 and 10.

$$y_i = (x_{i1})^1 + (x_{i2})^1 + ... + (x_{in})^1 \tag{1.1}$$

Each iteration through an entry in the dataset the actor will each raise, lower, or keep the value the same for each exponent. Then we use the equation (1.1) to evaluate how close it is the predicting the correct value of y. This is our loss function and the more it minimizes loss between iterations the more it is rewarded, since the explanations are getting 'better'. The logic here is that the equation roughly represents the weights of each feature in the prediction and that raising the exponent gives the feature more weight and lowering it does the opposite.

1.3 Preliminary Results

We used only one dataset for training this - soccer dataset
Preliminary testing of network produced pretty poor results. The final equations produced did a decent job of modeling the soccer data, but were not generally reliable. The equation given the features correctly modeled the prediction with accuracy around 60% using a margin of error of 5%. The goal is to have the final equation correctly model the predictions with a high degree of accuracy while still having the equation be easily readable and interpretable. Our linear equation is very readable, but not that accurate. For our final report, we will try to have reproducible and detailed test results.
The biggest limiting factor, we will try to address is the model is super linear with only modifiable

exponents. Trying to make a generalized readable equation is itself a huge challenge, but we probably can do better than linear. Trying to translate an equation to non math people is also a huge project for another time
Another limiting factor was our reward function only looking at the previous iteration when a better one would look at the sum of the rewards since beginning of training. Ideally we would like to test this on other datasets to see how it performs, so we are not over fitting to this particular soccer dataset;

2 Conformal Prediction

A key part in analyzing our results is quantifying uncertainty in our predictions. More specifically, understanding the confidence in point predictions of new examples.
While standard Machine Learning techniques generate a prediction model, there's no standard concept of confidence and credibility in prediction. To solve this issue, we've developed a general technique using conformal prediction that quantifies confidence in point predictions on a simple 3 layer Neural Network.

2.1 Theory

The two main goals of conformal prediction are as follows:

1. Confidence of prediction for new label.
2. Informativeness of each new data point.

Conformal prediction isn't an algorithm, instead a framework which implies that this technique can be bootstrapped on any ML model, making it quite useful. Historically, this technique has mostly been applied to regression tasks for determining confidence intervals. Other techniques such as Gaussian Discriminant Analysis (GDA) and other Bayesian frameworks tackle the same topic, but they require access to prior information, which isn't always the case in real-world.
Conformal prediction relies on the assumption that joint probability distribution should be the same. This can be proven via the test of exchangeability where every IID sequence is exchangeable (converse doesn't hold true). We're using a loans dataset to predict if a customer deserves a loan or not based on 11 features such as education status, martial status, and number of dependents for example. Since the dataset is sampled randomly from 484 individuals, we can use the central limit theorem to establish IID, thereby also proving exchangeability.

2.2 Approach

The methodology to implement conformal prediction can be broken down into 3 steps (assuming model has already been trained before and data is accessible for testing):

1. Calculate nonconformal score using Nearest Centroid algorithm
2. Calculate p-values corresponding to the current possible prediction/label
3. Output j as predicted label of the current example with p-value p_j if and only if $p_j \geq \epsilon$

2.3 Nearest Centroid Algorithm

A non conformal score represents how dissimilar a current object is with a group of objects of the same class, y_i. The higher the conformal score, the greater the dissimilarity. This can be solved using common techniques in information retrieval domain such as Knn or clustering techniques such as K-means and Gaussian mixture models. However, since the task deals with pre-labelled classes, we don't need to perform clustering in the traditional sense. Moreover, Knn is a computationally very expensive task as it needs to calculate the distance between all objects. This can be proven by calculating the non-conformity score using a 1 class K-nn:

$$\alpha_i^y = \frac{\sum_{j=1}^{k} D_{ij}^{y}}{\sum_{j=1}^{k} D_{ij}^{-y}} \tag{2.3.1}$$

As shown here, this model essentially computes the distance on the combination of all sets of objects, thereby running in O(n!) time. Even though this may be a reliable technique, we can save some time

by implementing a nearest centroid technique, which is similar to K-means clustering.
A nearest centroid essentially finds the uniform weighted average point in the distribution of some objects in class y_i. After finding the average point, we can label the incoming test point based on the proximity of the test point to the nearest centroid. Therefore, the non-conformity score using Nearest Centroid (NC) algorithm is shown as follows:

$$\alpha_i = \alpha_i(x_i, y_i) = \frac{d(\mu_{y_i}, x_i)}{min_{y \neq y_i} d(\mu_y, x_i)} \tag{2.3.2}$$

Here, μ_y represents the centroid for class y and $d(...)$ represents some distance function. We've chosen euclidean distance as a baseline, but cosine similarity and Jaccard Index can be used for more text-based prediction techniques.
As shown in Equation (2.3.2), we've been able to compute the non-conformity score from $O(n!)$ to $O(m)$ where m represents the number of classes in our data. We've implemented a more specific non-conformity measure for binary classification which is shown below:

$$\alpha_i = \alpha_i(x_i, y_i) = \frac{d(\mu_{y_i}, x_i)}{d(\mu_y^{1 \oplus y_i}, x_i)} \tag{2.3.3}$$

Since this is a binary classification problem, $\mu_y^{1 \oplus y_i}$ only deals with one of the two classes. For example, if μ_{y_i} is chosen, the opposite label will be chosen for μ_y.

2.4 Significance and p-value

In order to make use of the non-conformity score generated above, we need to introduce the notion of p-value which can be represented as follows:

$$p - value_i = \frac{count(j = 1, ..., n : \alpha_j \geq \alpha_n)}{n} \tag{2.4.1}$$

Unlike the standard definition of p-value in statistics, this measure essentially provides the ratio of non-conformity scores for a test point less than or equal to itself across the dataset provided. The higher the p-value, the lower the non-conformity score. This should be obvious because a small non-conformity score shows higher correlation between a test point and existing dataset. Using this measure, we can find the confidence of our predictions using $1-$ p-value$_i$. This will output a list of confidence scores ranging from $[0, 1]$ and we can use this metric to predict the accuracy in our test predictions based on ranges of confidence scores set by a parameter ϵ.

3 Results and Experiment

We explored a number of different common machine learning tasks in order to test the robustness of the conformal framework. The algorithms we used are: Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), Feed Forward Neural Network (ANN), and Deep Q Network with Prioritized Experience Replay buffers (DQN).

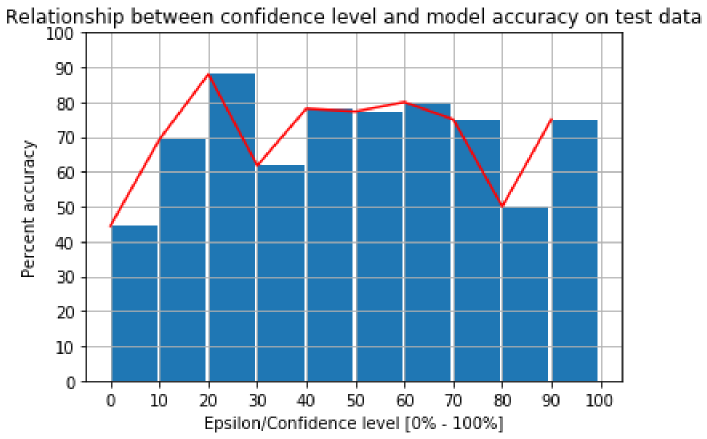
3.1 Recurrent Neural Network

In order to explore text processing tasks, we sought to classify spam emails. We felt that an Recurrent Neural Network would be best suited to this task because of its ability to make strong predictions on sequential data.

We removed stop words and utilized existing word embeddings for our data. Then we applied a basic LTSM with Cross-Entropy loss and Adam optimizer, using mini-batches. Our validation loss approached 0.5 over 10 epochs and we achieved a test accuracy of roughly 72 percent. Unfortunately, due to lack of time, we weren't able to apply Conformal Prediction framework on this particular algorithm. But we did so in the other 3 discussed below.

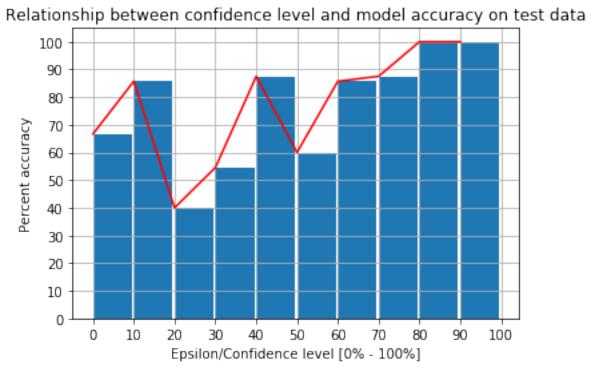
3.2 Convolutional Neural Network

For Image classification, we attempted to classify whether a person was COVID19 positive based on an CT-Scan image. For this problem we applied a Convolutional Neural Network because of their ability to extract patterns from structured data. We applied a basic CNN with 2 convolutional layers to our data. Our validation loss approached 0.56 over 30 epochs and we found a test accuracy of roughly 62 percent. After applying conformal prediction on CNN for COVID classification, we noticed an improvement of about 10% in test predictions when prediction confidence was between 30-70%. This can be demonstrated by the figure below. We can observe an approximate upward trend in percent accuracy as confidence increase for test predictions.

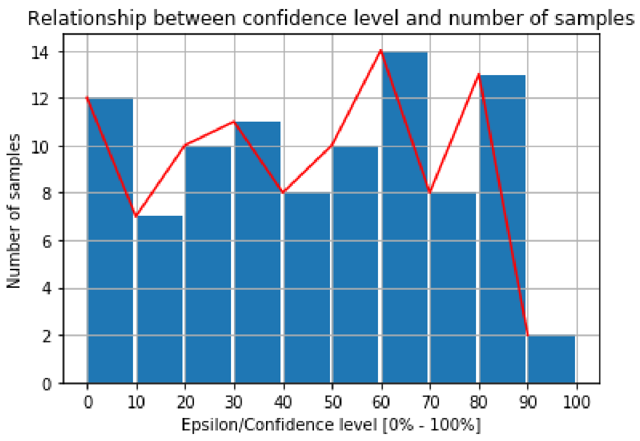


3.3 Feed Forward Neural Network

After implementing a simple 3 layer ANN on loans dataset (all code and data is provided in our GitHub, see references) using cross-entropy as loss function with Adam as our regularization algorithm, we found the loss to approach almost 0.2 after 11,249 epochs which took approximately 8 seconds to train.
Applying conformal prediction onto this model gave very good results, as shown in the figure below.

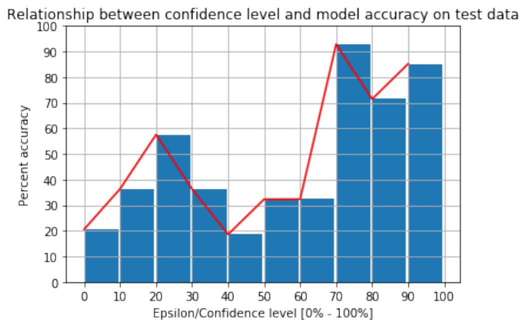


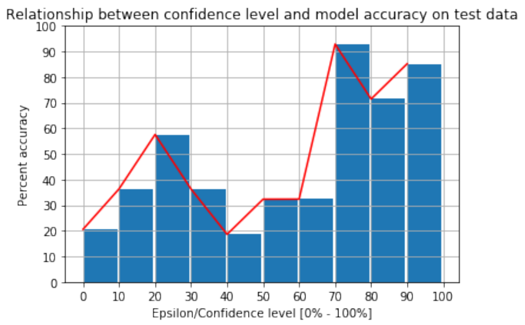
It is quite important to note the assumption that might occur from the above chart that the model might perform better due to shortage of data as confidence increases. This can be nullified with the chart shown below.



3.4 Deep Q Network

To really set us apart from all the projects, we took the ambitious leap by integrating our model with any imaginable network architecture possible. One suggestion was to incorporate an RL framework and test the reliability of Conformal Prediction on such a model. To accomplish this, we designed and developed a Double Deep Q Network that can solve the Lunar Lander environment from OpenAI gym in 2000 episodes (30 minutes of training time). Since RL is a different paradigm, the way we measured "truth" in the context of conformity was by creating 2 agents, and expert/behavior agent that would be the model of the program, and a supoptimal/evaluation agent to measure performance for. Both models were trained using delayed DQN with prioritized experience replay buffers. However, the optimal agent (behavior) was trained until full model convergence, while the evaluation agent was stopped prematurely.
To calculate non-conformity scores and p-values, we ran an episode of the model based on the states observed from the expert policy. We split the trajectories generated from the state space from both agents, while only the expert policy controlled the environment. From this, we were able to generate a feature matrix, and the problem now became that of a naive supervised setup that was demonstrated in the case of feed-forward neural network (3.3). After performing testing, we were quite surprised that this trend of model accuracy continued to exist as confidence levels increased. This can be summarized in the 2 figures shown below for 2 different evaluation (suboptimal) agents with expert rollout from behavior agent as described above.



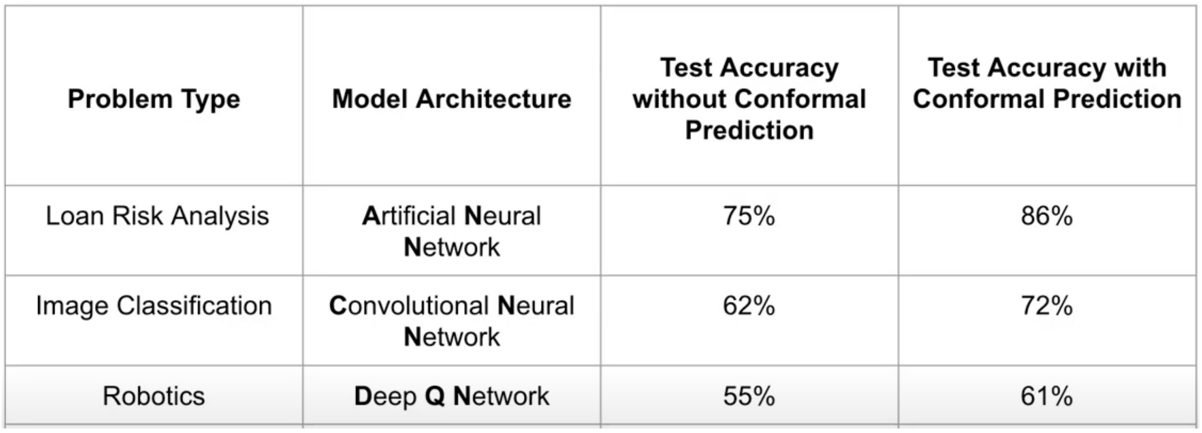


4 Limitations and Future Work

We started off with a broad scope and general idea of how we would implement our goals, but getting down into the details we ran into lots of challenges building custom environments for A3C, defining custom reward and loss functions etc. Given the time constraints and the fact we had to teach ourselves these new ML algorithms and Tensorflow and PyTorch, we had to reduce the scope of our project. This will adversely impact results as they will not be as ’good’ as expected. It was still a tremendous learning process.
That being said, we are incredibly excited of the results generated by our project and are confident that given the time frame, we far exceeded any expectations set by the department! This is a great

area of research, and we're personally committed in developing more robust and better models to advance uncertainty quantification in the positive direction.

To summarize, our results can be summarized in the figure below:



Problem Type	**Model Architecture**	**Test Accuracy without Conformal Prediction**	**Test Accuracy with Conformal Prediction**
Loan Risk Analysis	**A**rtificial **N**eural **N**etwork	75%	86%
Image Classification	**C**onvolutional **N**eural **N**etwork	62%	72%
Robotics	**D**eep **Q** **N**etwork	55%	61%

5 Contributions

Connor - Developed Actor-Critic network for model interpretability.

Qasim - Developed Conformal Prediction framework, developed ANN and DQN models, and implemented conformal prediction for CNN, ANN, and DQN architectures.

Joseph - Developed CNN and RNN models.

6 References

1. Conformal Prediction code for ANN, RNN, DQN, and CNN - GitHub
2. Conformal Prediction Theory - Science Direct
3. Actor Critic Networks - Google Colab notebook