Siemens AI Challenge
Scalable Voting Approach based on Genetic Algorithms

Alexandridis Konstantinos Panagiotis, Marco Fontana
Centre for Doctoral Training in Distributed Algorithms
University of Liverpool

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Siemens AI Challenge
Scalable Voting Approach based on Genetic Algorithms

***Abstract*—We propose the use of a voting system for classification and the PAC-Bayes theorem to bound the mis-classification error. Our system leverages the power of genetic algorithms to create a set of expert models that are different from each other due to genetic operations and very fit for the task of classification. Our system is scalable because instead of using all the expert models to make a new prediction, it uses a small artificial neural network (ANN). This network captures the knowledge of the complex voting system and imitates the voting system's output. The result is a very fast network that emulates the behaviour of the voting system. Our findings suggest that the ANN is better in performance than using a single optimised model, it is faster than using a classic voting system and it has a lower risk of error.**

Code available at: https://github.com/kostas1515/siemans_ai_competition

I. Introduction

Recent advancements in the field of machine learning and artificial intelligence have improve the quality of life for many people across the world. Applications like drug discovery, or lab automation are some examples were machine learning can benefit the health industry. Other applications such as autonomous vehicles, can benefit the transportation industry. Finally, smart voice assistants, chat bots and mobile phone apps make our lives easier by giving us information before even asking for it.Some sort of artificial intelligence already exists in our daily lives without even noticing it and even though we can enjoy the benefits it provides, we should also be very aware about the risks it poses in security, personal privacy and safety. As developers and consumers of these technologies, we ought to be ethical and consider not only the benefits but also the risks of these technologies. We think that the Siemens AI challenge is a great example, to tackle the difficulties and put our knowledge to the test, when it comes to safety critical applications. We believe that this challenge is very relevant in this period of time, when most things are done digitally and they do not always require human intervention.

II. Problem Definition

This challenge, gives three target datasets (D_A,D_B,D_C) and asks from the participants to develop a system that not only makes good predictions but also provides an upper bound for the misclassification error. This challenge is relevant in safety critical applications and requires the construction of autonomous systems that take decisions and minimise the risk of error.

To give more details, the problem is a simple classification problem that requires a system that distinguishes the green from the blue points. These points can be represented as $X = [x_0, x_1], Y = -1, 1$, where $x_0, x_1 \in [0, 1]$ can be thought as cartesian coordinates that belong to a unit square and y can be thought as the class and which is either -1 or 1 (green or blue).

Figures 1, 2, 3 are scatterplots of the datasets A,B,C respectively. Given these we are requested to build a classifier that should be able to assign a new data point $\bar{p}$ into the correct category. Furthermore, it should also provide an upper bound for misclassification error. Finally, the solution should be scalable with respect to more data and or dimensions. In the next sections we will tackle these problems and explain the methodology and the logic behind our ideas.

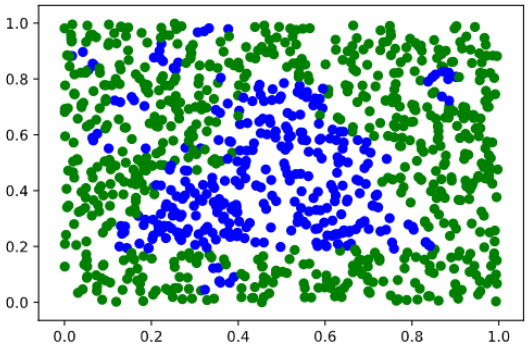

Fig. 1. Scatterplot of dataset D_A. Blue dots correspond to class $c = 1$, while the green dots correspond to class $c = 0$

III. Proposed Solution

From the scatterplots, we can observe that the datasets contain different quantities of data points, (D_A contains 1000 points, D_B contains 5000 points and D_C contains 10000 points). Also, datasets A and B are balanced with respect to class frequency, while dataset C displays a big imbalance. Finally, we can also notice that the data distribution is different for each dataset. Based on the above, we will develop three distinct classifiers, (one for each dataset).

To optimise the classification performance we suggest the use of genetic algorithms. To combat the complexity of each dataset we deploy a voting system based on the genetic algorithms. To combat the scalability aspect we distill the

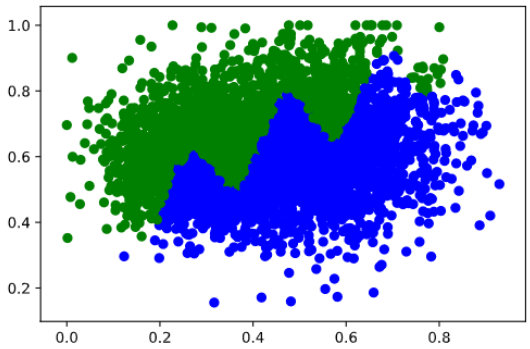

Fig. 2. Scatterplot of dataset D_B. Blue dots correspond to class $c = 1$, while the green dots correspond to class $c = 0$

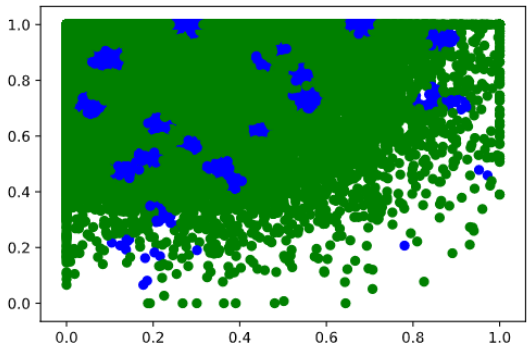

Fig. 3. Scatterplot of dataset D_C. Blue dots correspond to class $c = 1$, while the green dots correspond to class $c = 0$

knowledge of our voting system to a small artificial neural network (ANN), that is used to do real time inference. Finally, to compute the upper bound for misclassification error we will use the framework introduced by McAllaster in [1] In summary our system, is a ANN that has learned to imitate the output of a complex voting system of classifiers. These classifiers were obtained by using genetic algorithms in order to ensure that they are different from each other but also very optimal for the task. To support our results theoretically we use the "PAC-Bayesian" theoretical framework [1], and provide the risk of mis-classification in comparison to the risk of Gibbs classifier.

IV. Theoretical analysis for Miss-classification error

The backbone of our pipeline uses a set of classifiers called *majority votes classifiers*. They combine the predictions of several independent experts to achieve better classification performance based on the weighted sum of the independent votes. If the combination of the votes is deterministic, the algorithm approximates Bayes classifier, while if the output is determined by a stochastic distribution, the algorithm can be considered as a Gibbs classifier. Our approach is based on a deterministic combination of the results provided by the voting system. To obtain better results, we value every vote of the voting system differently, according to the performance of the "voter-classifier" in the test set.

The main result of the McAllaster framework is the "PAC-Bayesian Theorem", which bounds the risk of a Q-weighted majority vote (Bayes classifier) by bounding the risk of the associated Gibbs classifier. In this context Q is defined as the posterior distribution over the space of classifier that considers the information provided after the training of the algorithm.

A. PAC-Bayesian bound

In the proposed classification problem, we aim to determine the input-output pair (x, y), where $x = (x_1, x_2)$ belongs to the real-valued input space $\mathcal{X} = [0, 1] \times [0, 1]$, and y belongs to the discrete space of the categories, $y \in \mathcal{Y} = \{-1, 1\}$. Each expert (or voter) expresses the result of the binary classification by the function $e(x) : \mathcal{X} \to \mathcal{Y}$.

Suppose the training of the genetic algorithms provides a set of weights $f1^i$ on the votes $e_i(x)$, $i \in \mathcal{H}$ space of the voters, which defines a Q-weighted majority vote classifier. Given any $x \in \mathcal{X}$, the output of the classifier can be expressed by

$$B_Q(x) = \text{sign}[\mathbf{E}_{e\sim Q} e(x)] \tag{1}$$

We assume that the input-output pairs (x, y) are drawn i.i.d. according to the distribution D. We can define the the Bayes risk $R_D(B_Q)$, also called risk of the majority vote, as as the expected loss of the majority vote classifier B_Q relative to the distribution D on $\mathcal{X} \times \mathcal{Y}$.

$$R_D(B_Q) = \mathbf{E}_{(x,y)\sim D} I(B_Q(x) \neq y) \tag{2}$$

where $I(a) = 1$ if predicate a is true and 0 otherwise. As said previously, we can use the stochastic equivalent of the Bayes classifier to define a bound on the Bayes risk. We call *Gibbs classifier* G_Q the stochastic classifier that randomly chooses a voter, identified with e, according to the distribution Q and it returns $e(x)$ as result of the classification problem. The Gibbs risk corresponds to the probability that G_Q misclassifies an example (x, y) of distribution D:

$$R_D(G_Q) = \mathbf{E}_{(x,y)\sim D} \mathbf{E}_{e\sim Q} I(e(x) \neq y) \tag{3}$$

It is known that the risk of the (deterministic) majority vote classifier is upper-bounded by twice the risk of the associated (stochastic) Gibbs classifier [3].

$$R_{\langle(x,y)\rangle}(B_Q) \leq 2R_{\langle(x,y)\rangle}(G_Q) \tag{4}$$

V. Theoretical analysis of classification system

In this section we will describe the details of the proposed classification system. This system is built in three steps, first we train multiple models using genetic algorithms and create a set of expert models. In the second stage, we sample random datapoints from the uniform distribution and create a synthetic dataset D_S. Next, every expert model makes predictions on this synthetic dataset. We call these predictions, votes. Naturally, our expert models will agree in the easy datapoints and will vote for the same class, while they will disagree for the

ambiguous datapoints. We can capture the rate of agreement by taking the weighted mean of the votes for every datapoint. The hypothesis is that the rate of agreement will be proportional to the difficulty in predicting this datapoint. This difficulty can be also seen as a proportionality of error. Now that we have the set of expert models, it's becoming apparent that for a new datapoint we would have to consult all experts to get the final results. This is very difficult and does not scale well. Instead of this, we can model the vote agreement by training a small neural network. This network captures the knowledge from all expert models and learns how to evaluate the ambiguity of a new datapoint. The end result is that for a new datapoint the model produces two values one that represents the confidence for the prediction being class 0 and another for class 1. Finally we can compute the mis-classification risk of the ANN and compare it with the Gibbs risk to set the mis-classification upper bound.

Now that the main idea was presented, we can continue with explaining the implementation process in detail. As described our system development contains three steps. The first step is the genetic algorithms deployment, the second is the synthetic dataset creation and the third is the artificial neural network.

A. Genetic algorithms

We use genetic algorithms for two reasons, first to discover good classifiers (experts) and second to ensure that these experts are dissimilar from one another. Genetic algorithms are inspired from biology and use genetic operations such as crossover and mutation in order to refresh the population of solutions. In each generation, every chromosome (or potential solution-classifier) produces children solutions based on the cross-over and mutation factors. The fitness function evaluates the chromosomes in in the end of each generation and acts as a natural selection for chromosomes that provide better solutions. These chromosomes survive the natural selection and continue to reproduce in the next generations. As we understand, the choice of fitness function is very critical to the end result.

1) Choice of fitness function: The challenge is mentioning that different mis-classification errors have different costs. As it explains, classifying green point as red in a traffic lights intersection will cause minor inconvenience but classifying a red point as green can cause a serious accident. A utility function based on costs could be a valid fitness function but as the costs are not explicitly stated by the organisers, we choose a more standard fitness function which is the $f1$ measure (Equation 5) . The $f1$ measure is the harmonic mean of precision and recall, and it aims in producing solutions that are better in predicting the correct class while making plenty predictions. After all, we would not like a system that always predicts red just because the green prediction is too risky.

$$F1 = 2\frac{precision * recall}{precision + recall} = \frac{tp}{1/2(fp + fn) + tp} \tag{5}$$

where tp are the true positives, fp are the false positives and fn are the false negatives.

B. Synthetic dataset generation

In this step, we can use the experts models to generate a synthetic dataset. To do so, we sample datapoints from the uniform distribution and perform predictions for every datapoint and for every expert model. Since every expert model has different strength in it's predictions we will multiply the $f1$ measure with the prediction. In other words, the vote of the expert e_i, $i \in \{1 \dots n\}$ index among the n voters, for a synthetic datapoint p will be weighted by it's $f1_i$ measure:

$$\begin{aligned} V(e_i, p) = (1 - e_i(p)) * f1_i + (e_i(p))(1 - f1_i) - \\ e_i(p) * f1_i - (1 - e_i(p))(1 - f1_i) \end{aligned} \tag{6}$$

where, $p \sim \mathcal{U}$, $e(p)$ is the expert's prediction and can be either 0 or 1, and $V \in [-1, 1]$. V can be also expressed in one-hot encoding as in Eq.7:

$$\begin{aligned} V_{onehot}(e, p) = [(1 - e(p)) * f1_e + (e(p))(1 - f1_e), \\ e(p) * f1_e + (1 - e(p))(1 - f1_e)] \end{aligned} \tag{7}$$

After that we can gather the weighted mean of all the expert's votes in a synthetic dataset to represent the ambiguity of datapoints:

$$\bar{V}(p) = \frac{1}{G} \sum_{e=0}^{G} V(e, p) \tag{8}$$

where G is the number of generations and e is the finalist expert in the end of the generation.

In the next step we can use the synthetic dataset to distill the knowledge of our complex voting system to a small ANN.

C. Knowledge distillation from voting system to ANN

The main reason for developing the ANN is scalability. It is natural that if we keep adding an infinite number of classifiers into our voting system we can max out the performance, the scalability thought will suffer as for every new datapoint we would have to consult all the voters-expert models. To tackle this problem we train a small ANN that captures the knowledge of our voting system.

1) ANN architecture: We keep the complexity of the ANN very low. We design a fully-connected 6 layer feed-forward network (Figure 4) with leaky relu activations (Eq. 13) for the intermediate layers and sigmoid activation (Eq.10) for the output layer.

$$lr(z) = \begin{cases} z & \text{if } z \geq 0 \\ -0.1z & \text{if } z < 0 \end{cases} \tag{9}$$

$$\sigma(z) = \frac{1}{1 + exp(-z)} \tag{10}$$

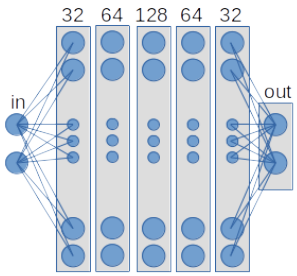

Fig. 4. Architecture of the artificial neural network. The numbers represent the numbers of neurons in each layer. The total number of trainable parameters for this small ANN is 21K.

2) Training details: We train our network using Gradient Descent with batch size equal to the whole dataset. We use Adam optimiser with a custom learning rate schedule that starts from 0.01 and reduces by a factor of 10 when it reaches certain milestones. We choose the Mean Squared Error (MSE) Eq.11 as our loss function because it is simple and easy to converge. Finally, we minimize the error between the one-hot encoded mean vote $\bar{V}_{onehot}$ and the network's logits Z.

$$MSE(Z, \bar{V}_{onehot}) = \frac{1}{N}\sum_{i=0}^{i=N}(\sigma(z^i) - \bar{v}^i_{onehot})^2 \qquad (11)$$

3) ANN output: We can represent the ANN as a function f that learns how to model the behaviour of the complex voting system. In more details, for a new data-point p it produces two values c_0 and c_1 that represent the confidence of that point belonging in class 0 and class 1 respectively. Note that $c_0 + c_1 \neq 1$ as these values are not probabilities.

$$f(p) = [c_0, c_1] \qquad (12)$$

D. Summary of proposed pipeline

In summary, the proposed pipeline starts with running the genetic algorithm to extract the set of expert-voters classifiers, (each classifier is discriminated by it's performance which is the $f1$ measure). In the next step the classifiers vote (with some strength which is the $f1$ measure) on synthetic datapoints. We gather the votes and take the mean according to Eq.8 and create a synthetic dataset. Finally we train a small ANN based on this synthetic dataset, in order to reduce the inference time of the complex voting system. The whole pipeline can be illustrated in Figure 5 . Next we will describe the empirical results that we obtained by using the proposed system on the three given datasets.

VI. Analysis and Interpretation

A. Genetic Algorithms training details

We deploy genetic algorithms based on TPOT framework in python to get the expert models.In our case we choose a mutation rate of 0.9 and cross-over rate of 0.1. We set the population size equal to 30 and we evolve for 100 generations.

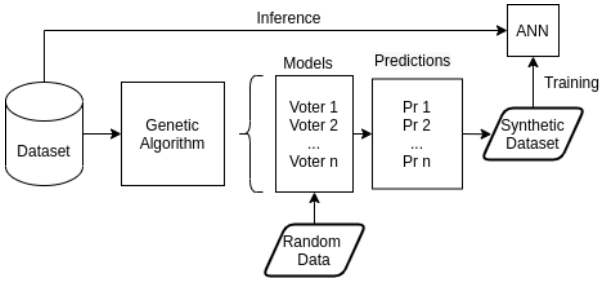

Fig. 5. Visualisation of the recommended pipeline

In the end of every generation we get the best finalist after doing 10-k cross validation on the validation set. This procedure gathers expert models that are dissimilar from each other, due to genetic operations, and they are incrementally better due to optimisation. We store these experts-models along with their achieved $f1$ measure and use them in the second step of the pipeline.

B. Synthetic dataset plots

We use the pool of experts obtained from the previous step to make predictions on 1 million random datapoints. We use Eq.6 and 8 to produce scatter-plots for the three datasets. As we can observe from the scatter-plots the mean is near zero in areas of high uncertainty, like near the decision boundaries Figures 6, 7, 8 or in extrapolated areas (see Figure 7 near the $[0, 0]$ or $[1, 1]$ coordinates.)

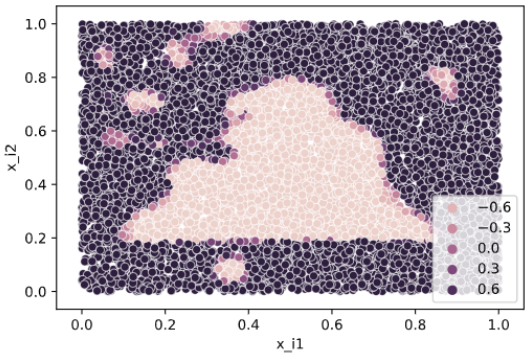

Fig. 6. Scatterplot of synthetic dataset D_{Sa}, which is the output of $\bar{V}(p), \forall p \in D_{Sa}$. Darker dots correspond to class $c = 0$, while the lighter dots correspond to class $c = 1$

In the next step we will use the generated synthetic datasets to train small ANNs that capture the knowledge of the voting system.

C. ANN output plots

We train the ANN according to the details outlined in section V-C2. To understand the ANN's behaviour we randomly sample 256^2 data-points from the uniform distribution and we pass them through the network. After that we probe c_0 output and we multiply it by 255 to create grey scale images.

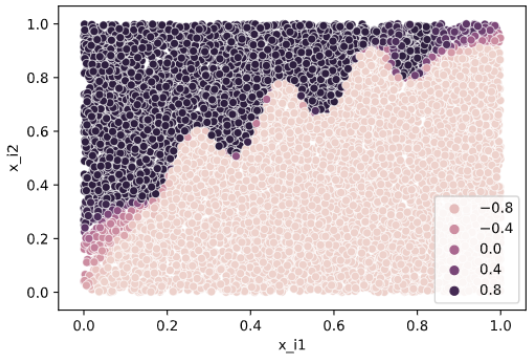

Fig. 7. Scatterplot of synthetic dataset D_{Sb}, which is the output of $\bar{V}(p), \forall p \in D_{Sb}$. Darker dots correspond to class $c = 0$, while the lighter dots correspond to class $c = 1$

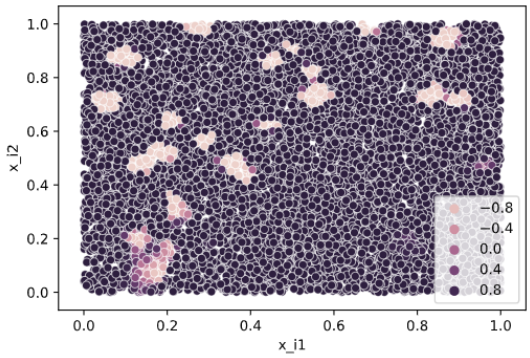

Fig. 8. Scatterplot of synthetic dataset D_{Sc}, which is the output of $\bar{V}(p), \forall p \in D_{Sc}$. Darker dots correspond to class $c = 0$, while the lighter dots correspond to class $c = 1$

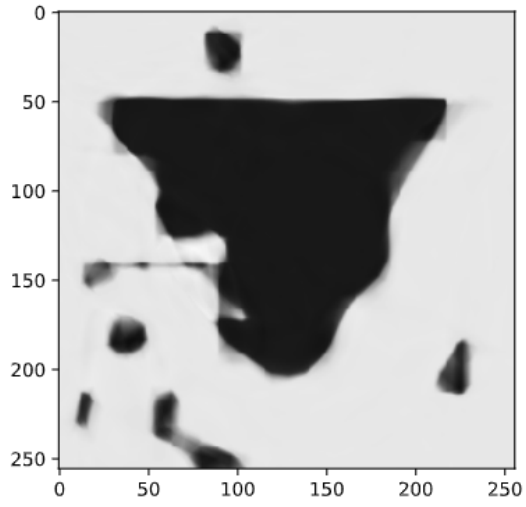

Fig. 9. Output of c_0 for the model trained on synthetic dataset D_{Sa}. Lighter colors are closer to 1 in amplitude while darker colors are closer to 0.

We can observe that the model can efficiently create boundaries and represent the ambiguity of the classes in the grey spectrum, lighter values are close to 1 in amplitude while darker are close to 0. Since the plots correspond to output c_0 the darker values would correlate with class 1 and the lighter with class 0. Apart from the difference in axis orientation, the images very closely resemble the synthetic datasets. This indicates that the ANN is able to accurately represent the complex voting system. Also, the ANN displays a high degree of smoothness in it's predictions which is a desirable property if one wants to extrapolate data outside of the unit square.

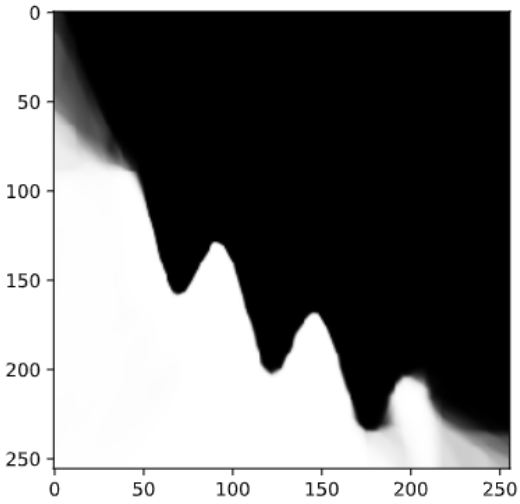

Fig. 10. Output of c_0 for the model trained on synthetic dataset D_{Sb}. Lighter colors are closer to 1 in amplitude while darker colors are closer to 0.

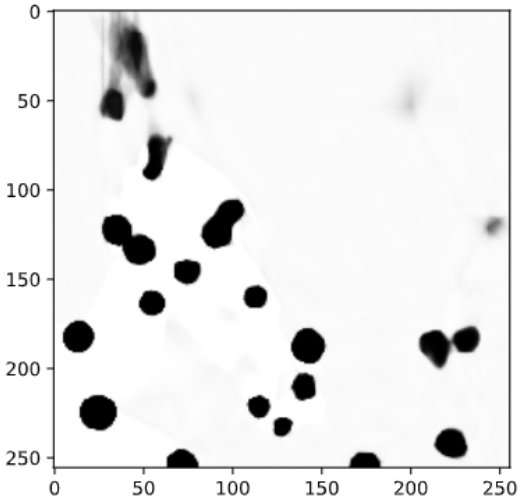

Fig. 11. Output of c_0 for the model trained on synthetic dataset D_{Sc}. Lighter colors are closer to 1 in amplitude while darker colors are closer to 0.

1) Strategies minimising the risk: The ANN produces two outputs c_0 and c_1 which is the confidence of a prediction being 0 or 1 respectively. In scenarios when the mis-classification cost differs, we can construct strategies to bias our predictions

TABLE I. PERFORMANCE $f1$

	Dataset A	Dataset B	Dataset C
Best Expert	0.9159	0.9953	0.9823
ANN	**0.9708**	**0.9981**	**0.9942**

based on that cost. For example we can choose the least risky option when the ambiguity of a datapoint is above a certain threshold. Mathematically, we can define the a strategy S to pick the least risky option c_0 if the datapoint p displays an ambiguity greater that a threshold t:

$$S(p) = \begin{cases} \max(c_0, c_1) & \text{if } |c_0 - c_1| \geq t \\ c_0 & \text{if } |c_0 - c_1| < t \end{cases} \quad (13)$$

In our approach we use $t = 0$ although our model can benefit from different values of t that suit different classification scenarios. In the next section we discuss the evaluation criteria.

VII. CRITERIA OF ASSESSMENT

We evaluate our method on three axis, on it's classification performance, on the bound of mis-classification error and the scalability towards more data and dimensions. As noted before, the $f1$ was used as a fitness function in the genetic algorithms step V-A Therefore, it is only natural to select this as the performance metric. Regarding the mis-classification error bound, we will compare the risk of the Gibbs classifier and the ANN as noted in Eq. 4. Finally for the scalability, we will measure the inference time of the ANN against the time of a single expert model.

A. Classification performance

As noted before, we report our results based on the $f1$ measure. We choose as a baseline the best model extracted from the final generation of the genetic algorithm. This model represents the best solution if we had to choose a single model instead of an combination of voting classifiers. To get a better approximation we do 10-k cross validation and take the mean of $f1$ measure as the final performance indicator. The results in Table I indicate that the ANN achieves better $f1$ measure than the best expert-classifier. From this we can infer two conclusions. Firstly, that the voting system is better than it's single counterparts. Secondly, that the ANN is able to imitate the behaviour of such complex voting system.

B. Empirical study of the mis-classification bound

In this section, we investigate the theoretical bounds derived in the previous section on the proposed classification algorithm. First we calculate the Gibbs risk. To do so, we use our voting system and randomly pick a vote as the final prediction. Using Eq.3 we calculate the Gibbs error.

Next, we compute the risk of the ANN and report the results. The results in Table III suggest that:

- the ANN has a lower risk than the Gibbs classifier
- the mis-classification bounds validate Eq. 4
- the ANN is a good approximation of the Bayes classifier

TABLE II. BOUNDS EVALUATION

	Dataset A	Dataset B	Dataset C
ANN Risk	0.06	0.004	0.0028
Gibbs Risk	0.06852	0.00772	0.003872

TABLE III. LATENCY (S)

	Dataset A	Dataset B	Dataset C
ANN-GPU	0.827	0.937	1.483
Expert-CPU	0.017	0.039	0.235

C. Scalability

In this section we discuss the merits of our approach with respect to scalability. The key concept to tackle scalability without compromising performance is the knowledge distillation to a small ANN. By making the problem more complex, towards data and dimensions the model's performance will not decrease as long as we keep adding more voting components. Once the desirable performance has been reached we can use knowledge distillation to transfer all the complexity into a fast shallow ANN. Our ANN processes a single datapoint in 0.48 seconds on an NVIDIA Quadro p4000, although it is able to process more datapoints in parallel and thus reduces latency of inference even further proportionally to the GPU memory. In Table III we can notice that the proposed ANN is slower than a random expert running on CPU. Even so, we notice that for bigger datasets e.g B,C the CPU will have a steeper increase in the computation time than that of the GPU ANN. Thus, we can infer that by increasing the amount of data there will be a point that the GPU ANN will outperform the CPU model in latency. Thus we conclude that the ANN is suitable and more scalable than a CPU counterpart.

VIII. CONCLUSIONS AND RECOMMENDATIONS

We presented a reliable and scalable solution to the proposed classification problem. Our approach is based on the genetic algorithms, which allow high classification performance and provide an evaluation of the confidence, given by the rate of disagreement between the experts. In order to achieve a higher degree of scalability, we distilled the uncertainty computed by the genetic algorithms in an ANN. With this approach, we avoid querying all the experts for each data point, speeding-up the execution of the classification algorithm. We proposed to evaluate the misclassification error of our approach by computing the PAC-Bayesian bound, defined on the stochastic equivalent classifier of our deterministic majority vote system. Unfortunately, the bound associated with the Gibbs (stochastic) classifier is often far from being tight to the misclassification error of a Bayes classifier, like the one we proposed in our solution. The reason is that the PAC-Bayesian bound doesn't fully consider the compensation of the misclassification errors of the single experts. A possible extension of our work would consider tighter bounds of the misdetection error, like the C-bound proposed by Germain et al. in [4]. Another extention

could be to train the same ANN only using the real data so that we can compare the benefits of the synthetic dataset generation and knowledge distillation.

APPENDIX

The theorem presented by Germain in [4] states that, under specific conditions, for any distribution Q on a set of voters and any distribution D on $\mathcal{X} \times \{-1, 1\}$, we have

$$R_D(B_Q) \leq \mathcal{C}_Q^D \tag{14}$$

where $\mathcal{C}_Q^D$ is the $\mathcal{C}$-bound. Germain et al. show that the $\mathcal{C}$-bound can be decreased by reducing the Gibbs risk or increasing the disagreement. It can be proved that the $\mathcal{C}$-bound can be arbitrary small by increasing the number of experts, under specific conditions.

REFERENCES

[1] D.A. McAllester. Some PAC-Bayesian Theorems. Machine Learning 37, 355–363 (1999). https://doi.org/10.1023/A:1007618624809

[2] D. A. McAllester. A PAC-Bayesian Tutorial with A Dropout Bound. McAllester, ArXiv abs/1307.2118 (2013)

[3] J. Langford, J. Shawe-Taylor. PAC-Bayes & margins. Advances in neural information processing systems. 2003:439-46.

[4] P. Germain, A. Lacasse, F. Laviolette, M. Marchand, and J.F. Roy. 2015. Risk bounds for the majority vote: from a PAC-Bayesian analysis to a learning algorithm. J. Mach. Learn. Res. 16, 1 (January 2015), 787–860.