**Proceedings of the [BAMC]**

**Best Applied Mathematics Conferences**

**BAMC 2025**

**January 15, 2025, Santa Clara, CA, USA**

**DRAFT - 00000.1**

**Enhancing Neural Network Performance using Belief Networks: A Combined Approach to Expert Evaluation and INHERITED Uncertainty**

**Robert Kleinhenz, Drazen Fabris, Christopher Waight, Varun Mangla**

Santa Clara University, Santa Clara, CA, USA

Abstract

*This research presents a method for creating belief networks given the classifications and certainty assigned to different labeled images in a truth set from an ensemble of experts. It further addresses the quality of these experts to create a belief network that models the certainty of each label in each image. This belief network is used to verify results from a Convolutional Neural Network classifying images in a similar domain. It demonstrates a priori evaluation of images, showing how uncertainty propagates through the truth set down to the labels. Furthermore, it demonstrates a priori assessment, adducts all possible solutions in the belief network, then propagates the uncertainty from all solutions to assign uncertainties to the CNN’s assessment. It presents two different algorithms for posteriori evaluations, the differences in performance and addresses issues such as the solution not being within the Belief Network’s feasible region.*

Keywords – Belief Networks, Uncertainty, Inheritance, Probabilistic Evaluations, Computer Vision

# **Introduction**

Convolutional Neural Networks (CNN’s are becoming more commonplace across all industries, with applications ranging from self-driving cars to early cancer detection with surprisingly high accuracy and recall [?]. However, performance of these CNN’s deteriorates when not enough training data is available [?]. In this instances, human experts are needed, but even expert opinions can differ. Furthermore, the availability of these experts might be limited, due to their availability, cost, or the size of the task might be an infeasible scale for human experts.

Previous work on this includes the development of Markov Monte Carlo Chains, one shot-learning, YOLO, and transfer learning techniques. Data augmentation and Generative adversarial techniques are also being investigated. These methods have problems such as introducing bias, etc etc.

In this paper, we gather an ensemble of experts, and rate their quality by giving them an assessment exam. Their quality is rated as a number + some standard deviation. Any experts whose Upper Confidence bound is above some agreed upon threshold are considered to be an expert. These experts are then asked to label several images in a truth set. They are also asked to give their own self confidence that the label they are providing is accurate. This dataset of different images with labels from experts along with the expert quality and self confidence score is combined to form a Belief Network.

We then use a CNN whose quality is unknown, due to a dearth of training information. Once this CNN classifies the image, the output is used to find all possible combinations and weightings of images and labels that can be combined that have the same expectation value as the CNN. Once all these combinations are discovered, the probability distributions of the BN are sampled, and the resulting histograms represent the inherited uncertainty of the Belief Network.

In this paper, we explain how the experts are assessed, and how they provide label, image pairs for each image in the dataset. We show how these uncertainty functions are combined in the event that more than one expert labels the same image with different certainties, or with different labels. We then show how these distributions can form different classifications with different weightings. We then show our method for determining all possible weightings of these distributions that give a classification with the same likelihood as an NN. We discuss our simulation results, the conclusions, and future work respectively.

# **Belief Network Creation**

**2.1** **Expert Evaluations** P̃

Each expert is shown an image and asked to provide a label. We compare it to the truth and treat it as a Bernoulli trial. We then can get a mean and confidence interval based on the results of their Bernoulli trials.

#(2)

**2.2 Weighted Self Assement = V\***

Once an expert is evaluated, they are asked to provide their self confidence in the answer V\*. This value must lie between 0 and 1.

P̃ and V\* are combined to make a triangular function as described in below and seen in figure 1.

A diagram of a function

Description automatically generated

**Figure 1: Density Function Shape**

In the image above the base of the triangle is chosen as mu-1.95\*std.dev, mean, mean+1.95std.dev). The height is chosen to make the area =1.

**2.3 Combining Multiple (label,image) Expert Evaluations**

We can combine multiple experts by doing convolutions. Note, triangular convolutions are difficult. These can be done via FFT, DFFT, or by flip-and-slide technique. In each of these methods, it involves turning our continuous distribution into a discrete one. The flip and slide has the finest resolution, so that is what we are going for.

A diagram of a diagram of a photo

Description automatically generated

**FIGURE 3:** Showing Convolutions being Collapsed into a single Distribution

The figure below shows how the triangular convolutions and clipped normal compare. (Figure shows something completely different needs to be updated.)

A diagram of a triangle function

AI-generated content may be incorrect.

**FIGURE 4:** Showing Convolutions being Collapsed into a single Distribution

**2.4 Creation of a Belief Network**

**2.4.1 Single Expert Belief Network**

In this section, we talk about how we combine all this together, and how it is well suited for matrix form.

An expert, E, is asked to looking an image Φ . The image Φ can be associated with up to M labels (**L**). The Expert provides a self certainty V\* that the image being viewed has a particular label on it.

We assume these labels are categorical, so if the expert is associating more than 1 label, the sum of their self certainties must be equal to 1. For example, an image of a 4 legged creature can be 0.59 cat, and 0.41 dog. However, a dog and cat will not be present in the same image. Image segmentation techniques can be used to ensure this.

For each of these connections between the image, the self certainty, V\* and the expert’s quality, P̃, are combined to form a triangular function *f*, as described in Section 2.2

This process is repeated for each image Φi from 1 to N in the set of images known as the **belief image set**.

This can be visualized as network, as demonstrated in figure 5.

A diagram of a diagram

AI-generated content may be incorrect.

**FIGURE 5:** A Belief Network as a graph

Using this visualization, one can represent the information compactly as a matrix, where each fij represents the distribution of image i from 1 to N, to each label j, from 1 to M.

A screenshot of a computer

AI-generated content may be incorrect.

**FIGURE 6:** The Belief Network being shown as a Matrix, with each f(ij) is a distribution instead of a scalar.

If a connection between a particular image and label is not made, it can be replaced in this matrix with a 0.

**2.4.2 Multiple Expert Belief Network**

A belief network can also be made to incorporate knowledge from multiple experts. Imagine there were K experts available.

Each expert, E, looks at the images in the belief image set, and assigns labels with self certainties. As in section 2.41, these connections and the expert’s quality score are used to create probability distributions are created for each connection. Each Expert ends up with it’s own F matrix.

Then, these matrices are then convolved elementwise.

F111 \* F311 \* F211 \* …. \* FK11

This is repeated for f11 though fNM. The result is a single belief network matrix **F,** which contains the insights of all the experts in a single compact form.

A white paper with black writing on it

AI-generated content may be incorrect.

**NB1.** We are not actually convolution with 0’s if they are present. In that case, we only do convolutions on none 0 entries.

**NB2**. In the event that an expert does not view a given image, then a column of 0’s should be added to that Experts Matrix prior to convolutions.

**2.5 A Priori Evaluations**

Imagine a new image Ψ. If we somehow knew its relative similarity φ, , to each image in the belief image set, we could create the following inference network. Note, sum of all φ must lie between 0 and 1.

A drawing of a diagram

AI-generated content may be incorrect.

**FIGURE 7:** The Belief Network being expanded into an inference network, as shown in Figure 3 of Bob’s jobs

Furthermore, we could determine the labels that would be associated with this image Ψ by performing matrix multiplication as below.

A graph on a white paper

AI-generated content may be incorrect.

However, since each fis distributions and not scalars, they must be sampled before use. Each f can be sampled repeatedly, and a histogram for the values of [l1, l2, … ,lm] can be built up. This would show the uncertainty of each label. This histograms could also be analyzed to quantify the uncertainty (not just a visual representation).

Here is a worked example. There are 10 experts, there accuracy scores are:



These 10 experts are shown 30 images, and they provide their predictions, as well as there self-certainty scores. Here are the 30 images:

A black and white numbers

AI-generated content may be incorrect.

For the most part, these experts agree, but here are some examples where they don’t agree.

A screenshot of numbers

AI-generated content may be incorrect.

A screenshot of a black screen

AI-generated content may be incorrect.

Nonetheless, belief networks for each Expert are made, and then combined as described in the previous section.

A new image is then introduced:

A white line in a black background

AI-generated content may be incorrect.

‘Somehow’ The relative importance is determined. In this instance, we use SSIM structural similarity index measure, a method for predicting the perceived quality of digital television and cinematic pictures, as well as other kinds of digital images and videos. It is also used for measuring the similarity between two images (see Wikipedia).

The photos in the figure below are the belief network set below are ranked and ordered by decreasing SSIM score.  
The SSIM scores are put in a vector, and the vector is normalized. The resulting vector in this example is:

A black background with white numbers

AI-generated content may be incorrect.

A number in black squares

AI-generated content may be incorrect.

By sampling F 1,000 times, and each time multiplying by the SSIM scores or the φ vector, we obtain the following histogram.

A graph with colored lines

AI-generated content may be incorrect.

NB. SSIM is a poor way of doing this. It is more commonly used to find duplicate photos, or near duplicate photos, and not for categorizing what is inside the image.   
  
A better way of coming up with a φ vector is to ask someone to look at the new image, and pick out a few images from the belief image set that they think look similar and assign some weights to it.

**2.6 Posteriori Evaluations**

The softmax output of a neural network, used for image categorization, provides a single scalar value for each label. This scalar represents the probability of the input image belonging to that specific category. While softmax outputs a probability distribution across all labels, it doesn't inherently provide a measure of uncertainty or confidence for individual labels.   
  
We posit that the softmax could be written as the vector L. Since we have the belief network in Matrix for (**F**), the problem of finding the similarity vector φ that satisfies the equation:

**F** φ = L

To do this, we use the mean value of each f*ij* function in the F Matrix. To account for the probability that an exact solution does not exist, we reframe the problem as finding a valid φ that satisfies:

**F** φ <= L

sum(φ) <=1

φ*i* >=0

The solution space for possible φ’s is infinite, so we suggest that the objective function should be to minimize:

|| L - **F** φ ||inf

This is to find the φ that generates an L^ with the smallest sup norm from the prescribed L. This is a Nonlinear objective function, which means that the solution is not restricted to a vertex.

A black screen with white text

AI-generated content may be incorrect.

However, the problem can be reformulated into a linear one. To do so, we rewrite the problem as:

A black background with white text

AI-generated content may be incorrect.

Putting this in standard linear optimization form, we write it as:

A black screen with white text

AI-generated content may be incorrect.

Finally, we can write this in the standard form of (Ax<=b) as:

Solve for x which minimizes the objective function c=t  
given:

A black background with white text

AI-generated content may be incorrect.

In this formulation, *the objective function exhibits linearity. This necessitates* that the optimal solution reside at a vertex of the augmented space defined by the matrix above. Specifically in our example, A corresponds to F, x represents φ (phi), and t denotes the magnitude of the supremum norm undergoing minimization.

A significant challenge arises from the fact that most linear programming algorithms yield only a single optimal solution. The identification of multiple optima, should they exist, presents a greater degree of complexity.

To address this, we employed Motzkin's algorithm for vertex enumeration. This procedure systematically identifies and stores all vertices of the feasible region. Subsequently, each vertex is evaluated to determine those that achieve the optimal objective function value. Due to the computationally intensive nature of this approach, the scope of our illustrative example is constrained to a belief set comprising 30 images with 10 possible labels.

All vertices that satisfy the optimum criteria are then averaged, and this found phi^ is used for inference as described in the apriori example.  
Below is the full algorithm.

Given L vector and F (Belief Network)

Initialize Lposteriori= []  
Find Favg  
Perform Linear Reformulation.  
Perform Vertex was on new space.  
Evaluate vertices and record optimal solutions.  
Average Optimal Solutions as Phi^  
Repeat A times (A🡪 inf):  
 Sample F  
 Calculate L^ = Fsampled\*Phi^  
 Append L^ to Lposteriori

Create a histogram of Lposteriori

The resulting histogram shows the closest possible Lposteriori to the Lgiven that the Belief Network can generate.

It also has an added feature of embedding the uncertainty of the BN into the softmax reading.

An example implementation is seen in the notebooks attached. Run BN\_Dat\_file\_Processor\_motzkin for an example.

**2.7 Posteriori Evaluations – Double Sampling**

Here is a version of the program that doesn’t just use the average F, but samples repeatedly from it.  
  
Initialize Lposteriori= []

Repear B Times (B🡪 inf)  
 Sample F  
 Perform Linear Reformulation.  
 Perform Vertex was on new space.  
 Evaluate vertices and record optimal solutions.  
 Average Optimal Solutions as Phi^  
 Repeat A times (A🡪 inf):  
 Use the same Fsample as outerloop  
 Sample from solution space of phi^  
 Calculate L^ = Fsample\*Phi^  
 Append L^ to Lposteriori

Create a histogram of Lposteriori

It is theorizes that these give equivalent histograms when A and B are large enough for the Central Limit Theorem to be used.

**3. RESULTS AND DISCUSSION**

We have some results…..

These results should show the results from sampling and double sampling.

**3.2 Future Work**

This type of problem is well suited for quantum computers. This promises to overcome the scalability issues with this. More work is needed to investigate.

**4. CONCLUSION**

This work successfully shows BN augmenting the usefuleness of CNN’s by allowing them to inherit uncertainty, making them more useful.

**5. ACKNOWLEDGEMENTS**

We acknowledge that while our work deserves a Nobel prize, we probably won’t get one.

**6. REFERENCES**

[1] All References need serious update. This was lifted Don Quixote Style.

[2]  Pantoja M, Fabris D., Behrouzi A. “Deep Learning Basic Overview” Concrete International Magazine September 2018

[3]  Tesla Crash Preliminary Report US department of transportation NHTSA PE 16-007

[4]  Sun S, Chen C, and Carin L Learning Structured Weight Uncertainty in Bayesian Neural Networks Pro- ceedings of the 20th International Conference on Artifi- cial Intelligence and Statistics (AISTATS) 2017 Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54

[5]  Kendall A and Gal Y What uncertainties do we need in Bayesian Deep Learning for Computer Vision NIPS 2017 https://arxiv.org/abs/1703.04977

[6]  Gal Y, and Ghahramani Z. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning Proceedings of The 33rd International Conference on Machine Learning PMLR 48:1050- 1059, 2016

[7]  Deceus T. Handling imprecise and uncertain class labels in classification and clustering .Bayesian Deep Learning COST Action IC 0702 Working group C, Mallorca, March 16 2009

[8]  What my Deep Learning model Doesn’t know Y Gal

[9]  www-compsci.swan.ac.uk\csphil\CS345\chapts5-9.pdf

[10]  Hackerman, David The Certainty-Factor Model, Encyclopedia of Artificial Intelligence Second Edition

Wiley, New York pp. 131-138

[11]  Pearl, Judea Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference Morgan Kauf- mann San Mateo CA

[12]  Klir, George J.(editor), Yuan, Bo(editor) Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems –Selected Papers by Lotfi A. Zadeh, Advances in Fuzzy Systems – Applications and Theory Vol 6 World Scientific

[13]  Knuth, D. E. The Art of Computer Programming, Vol 2, Section 4.3.3, pp 290-295

[14]  Press, W. H., et. al., Numerical Recipes in C, Section 8.10, pp 329-343.

[15]  Google Research Research Blog: AlphaGo: Mastering the ancient game of Go with Machine Learning. 27 January 2016

[16]  Alex Kendall and Vijay Badrinarayanan and Roberto Cipolla, Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding, CoRR, abs/1511.02680, 2015, http://arxiv.org/abs/1511.02680, arXiv, dblp computer science bibliography, https://dblp.org

[17]  Hendrik J. Weideman Quantifying Uncertainty in Neural Networks https://hjweide.github.io/quantifying- uncertainty-in-neural-networks

[18]  Avis, David and Fukuda, Komei A Pivoting Algorithm for Convex Hulls and Vertex Enumeration of Arrangements and Polyhedra Discrete & Computational Geometry 1992, Sep,01 v. 8,n. 3,pp.295–313