Uncertainty Quantification for Classification

Course: EE691 RnD

By: Sanika Padegaonkar (20D070069),
Simran Tanwar (20D070078)

PhD Mentor: Gouranga Bala

Guide: Prof. Amit Sethi

Link to code

Introduction

- **Uncertainty** refers to the lack of confidence for each output of a machine learning algorithm.
- Incorrect overconfident predictions can be harmful in critical use cases such as healthcare or autonomous vehicles.
- For safe and informed decisions, the models would not only have to provide an output, but also describe as accurately as possible the level of certainty in their outcomes.

Literature Review
Sources of Uncertainty in Machine Learning - A Statisticians' View

Aleatoric Uncertainty

- Uncertainty arising from the inherent randomness of an event
- Irreducible
- Var(Y|x)

Epistemic Uncertainty

- Uncertainty due to a lack of knowledge
- Reducible
- All the remaining uncertainty
- Further divided into- *Model* and *Parametric/Estimation* uncertainty

Literature Review
A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges

- **Bayesian Deep Learning/Bayesian Neural Networks**

This technique refers to extending standard networks with posterior inference in order to control over-fitting. From a broader perspective, the Bayesian approach uses the statistical methodology so that everything has a probability distribution attached to it, including model parameters (weights and biases in neural networks).

Literature Review
A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges

- **Monte Carlo (MC) Dropout**

It is difficult to compute the exact posterior inference, but it can be approximated. In this regard, Monte Carlo (MC) is an effective method.

MC Dropout uses dropout as a regularization term to compute the prediction uncertainty.

Literature Review
A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges

- **Markov Chain Monte Carlo (MCMC)**

It starts by taking a random draw z_0 from distribution $q(z_0)$ or $q(z_0|x)$. Then, it applies a stochastic transition to z_0, as follows- $Z_t \sim q(z_t|z_{t-1},x)$. This transition operator is chosen and repeated for T times, and the outcome, which is a random variable, converges in distribution to the exact posterior.

Drawbacks: Sufficient number of iterations T is unknown and it requires a long time to converge.

Problem Statement

Through our experiments, we aim to answer the question "Does softmax give the true prediction probabilities?"

If it does not, then how much can we trust a prediction i.e. what is the uncertainty associated with it?

We use **entropy** as a measure of uncertainty in both our experiments.

Experiments

1. Test Time Augmentation
2. Test Time Dropout

Training

- **TTA**
  - **Resnet18** was trained on the **MNIST** dataset for **15 epochs**.
  - The **best validation accuracy** for **MNIST** was **99.2%.**
  - **Resnet18** was trained on the **CIFAR10** dataset for **100 epochs**.
  - The **best accuracy** for **CIFAR10** was **81.59%.**
- **Test Time Dropout**
  - **Resnet18** was trained on the **MNIST** dataset for **10 epochs**.
  - The **best validation accuracy** for **MNIST** was **99.2%.**
  - **Resnet18** was trained on the **CIFAR10** dataset for **50 epochs**.
  - The **best accuracy** for **CIFAR10** was **75.55%.**

Test Time Augmentation (TTA)

- TTA performs random augmentations to the test images.
- Thus, instead of showing the regular, “clean” images, only once to the trained model, we will show it the augmented images several times. We will then average (or mode) the predictions of each corresponding image and take that as our final guess.

TTA- Augmentation

We augmented a single test sample 15 times and considered the mode of these 15 predictions as our final prediction.

The following transform was used:

```
# set transformation option
tta_transform = transforms.Compose([
        transforms.RandomAffine(degrees = 10),
        transforms.RandomPerspective(),
        transforms.RandomHorizontalFlip(p=0.5),
        transforms.GaussianBlur(kernel_size=5, sigma=1.5)
        ])
```

TTA- Entropy Calculation

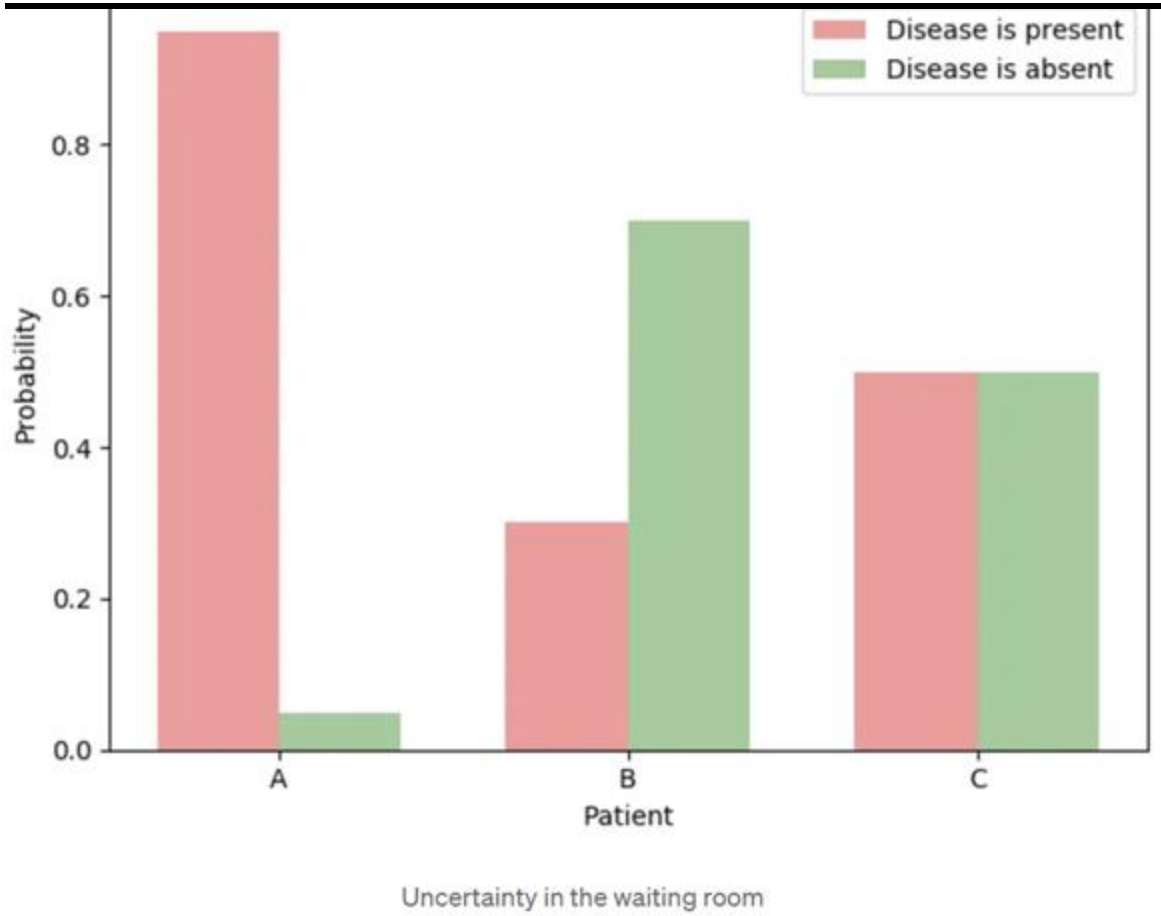
$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

Shannon entropy

```
[ ]  def entropy(aug_preds_all):
       entropies = []
       for aug_preds in tqdm(aug_preds_all):
         aug_preds = np.array(aug_preds)
         probs = softmax(aug_preds)
         log_probs = np.log2(probs)
         prods = np.multiply(probs,-log_probs)
         entropy = np.sum(prods)
         entropies.append(entropy)

       return entropies # returns entropies of all images in
```

Softmax was applied on the 15 predictions of augmented images, giving us 15 probabilities. Entropy was calculated using these probabilities.

TTA- Dropping samples

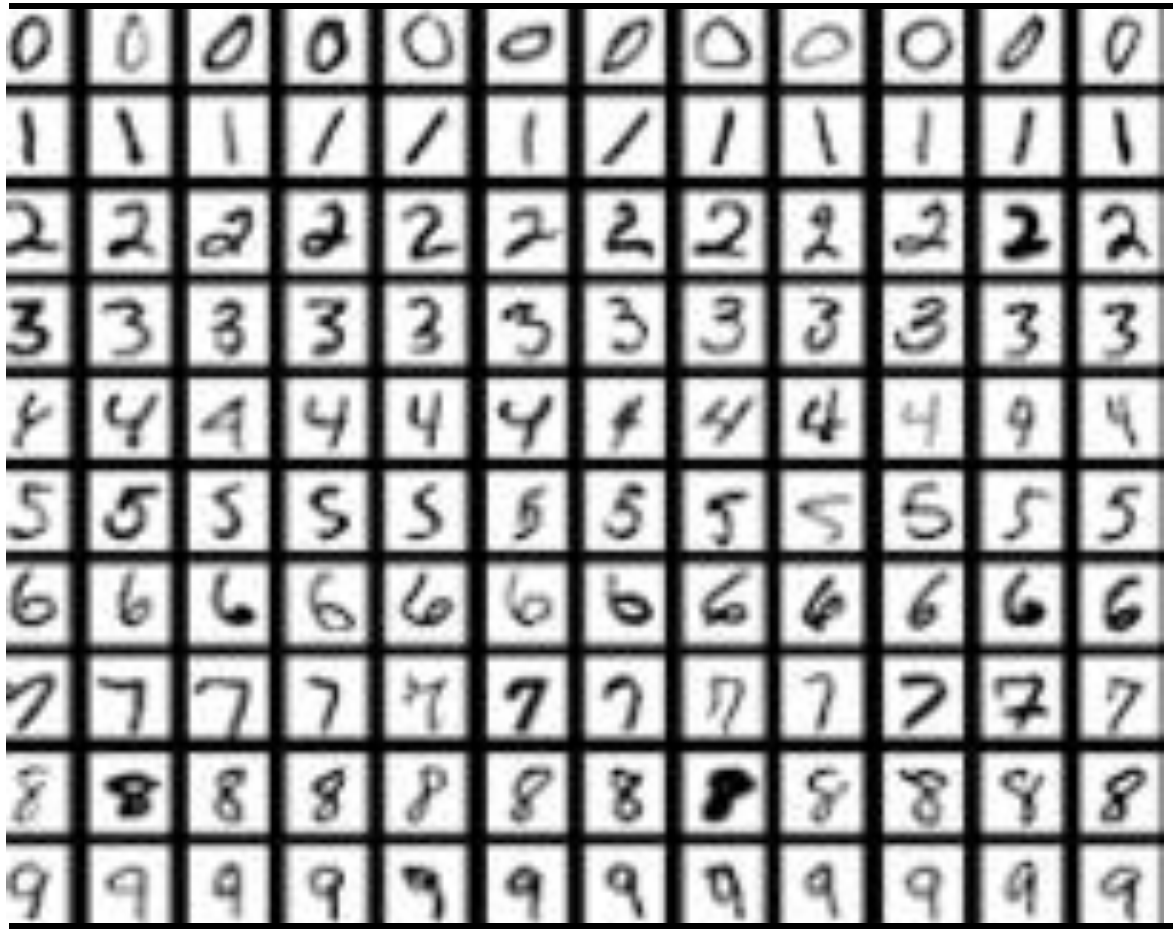


- In our case, a prediction with ***high*** entropy is a good prediction!

TTA- Dropping samples

- We dropout a certain fraction of the test set that contains "bad predictions" i.e. images have **lower entropy** than the others.
- To do this, we arrange the images in ascending order of their entropies and drop the first few rows depending on the drop rate.
- Thus, we increase the drop rate from 10% to 90% and expect the accuracy to increase as the drop rate increases.

TTA on MNIST

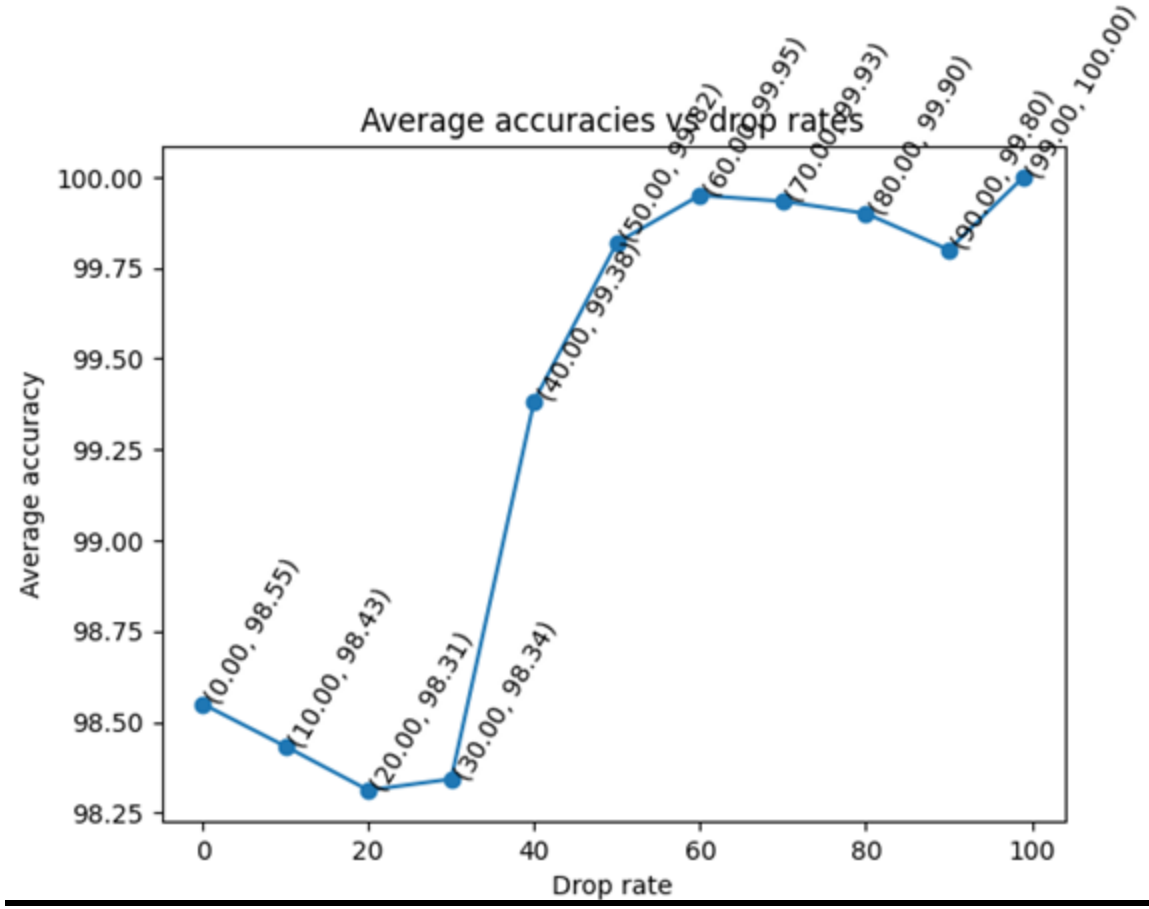


The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples.

TTA on MNIST

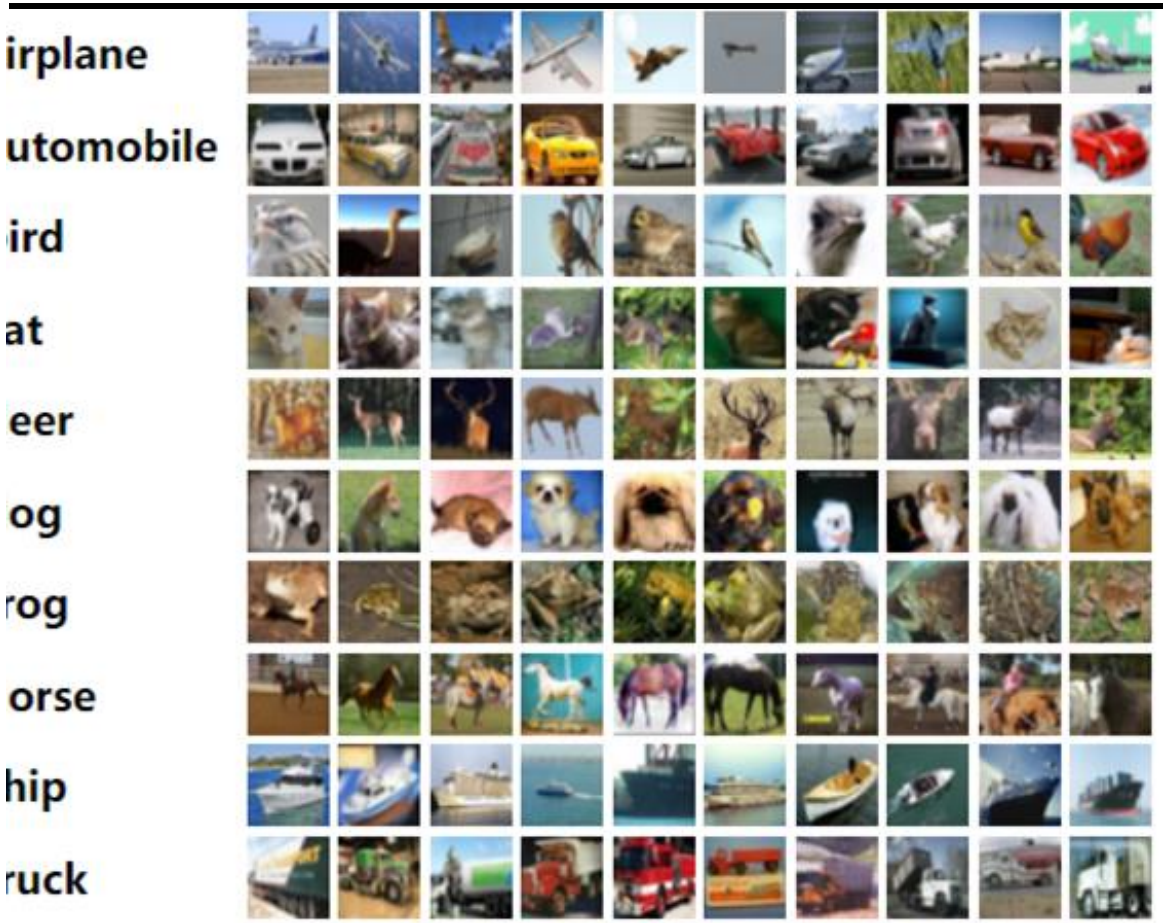
Column1	Image label (GT)	Pred Label (TTA Mode)	TTA Accuracy	Entropy
0	7	7	1	3.321928095
1	2	2	1	3.321928095
2	1	1	1	3.321928095
3	0	0	0.5	1.408725115
4	4	4	0.9	0.497363117
5	1	1	0.9	0.220725505
6	4	4	0.9	1.873393561

TTA on MNIST



The accuracy, averaged over all predictions, increases with increase in drop rate.

TTA on CIFAR10

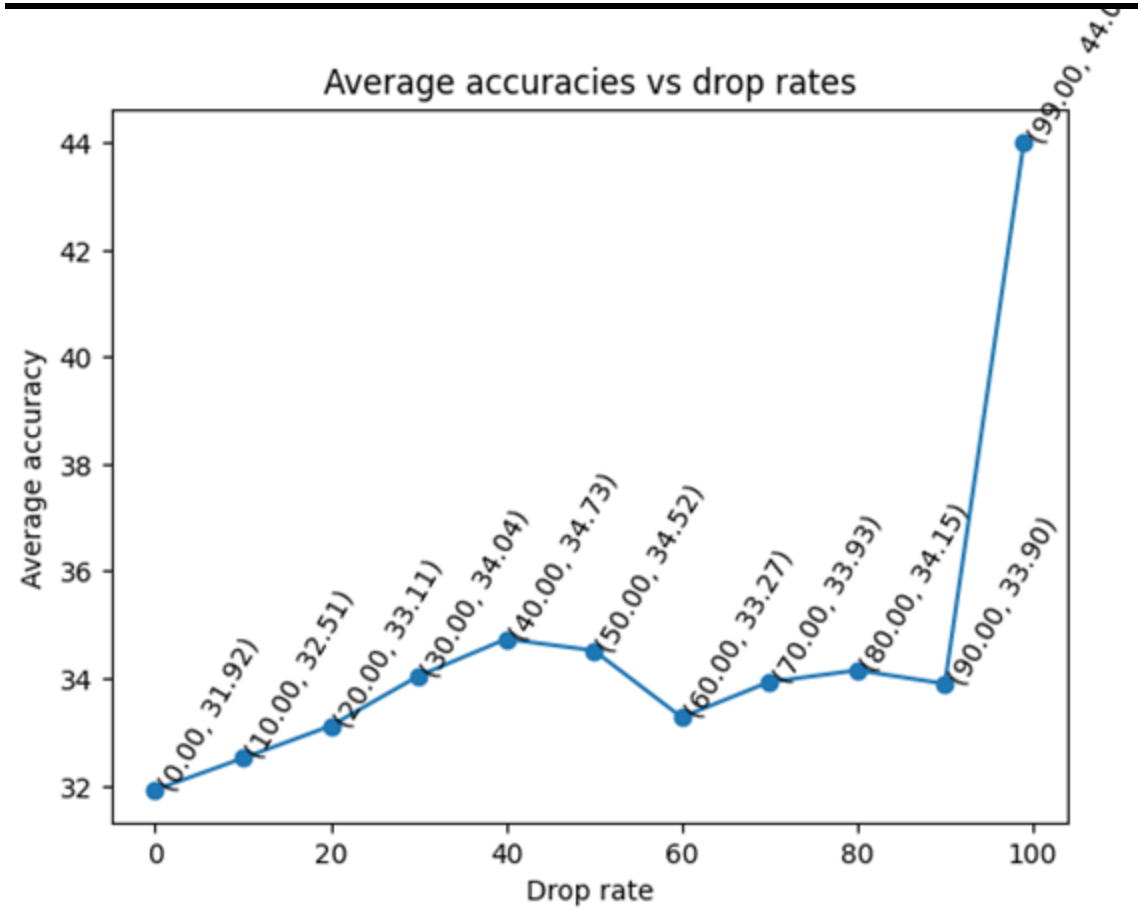


The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

TTA on CIFAR10

Column1	Image label (GT)	Pred Label (TTA Mode)	TTA Accuracy	Entropy
0	3	3	0.9	2.733773913
1	8	8	0.7	2.827552947
2	8	8	0.7	3.017388
3	0	8	0.5	2.326282448
4	6	4	0	3.284377577
5	6	3	0	3.321928095
6	1	3	0	3.321928095

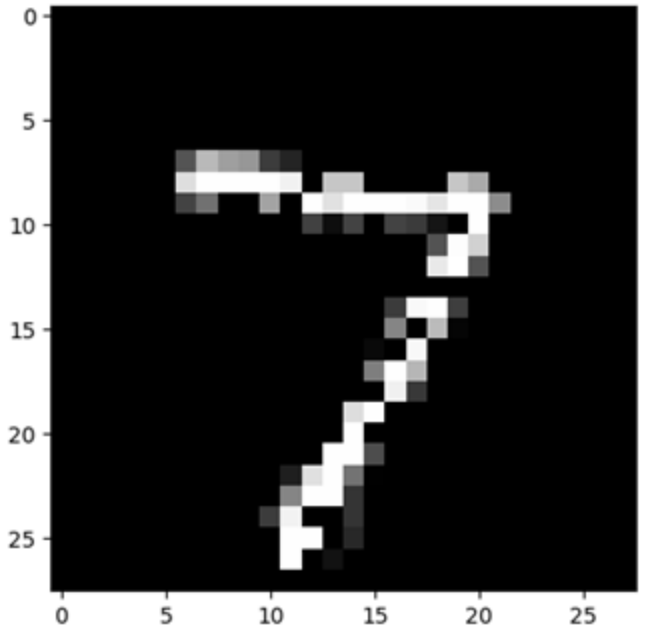
TTA on CIFAR10

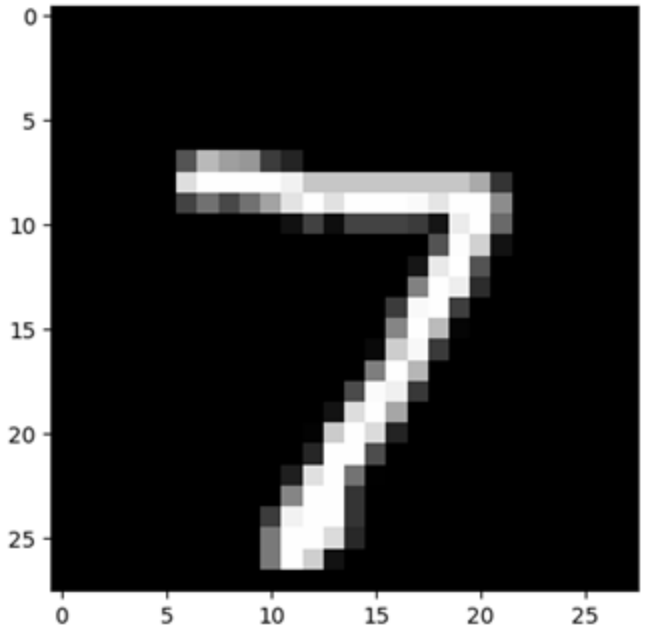


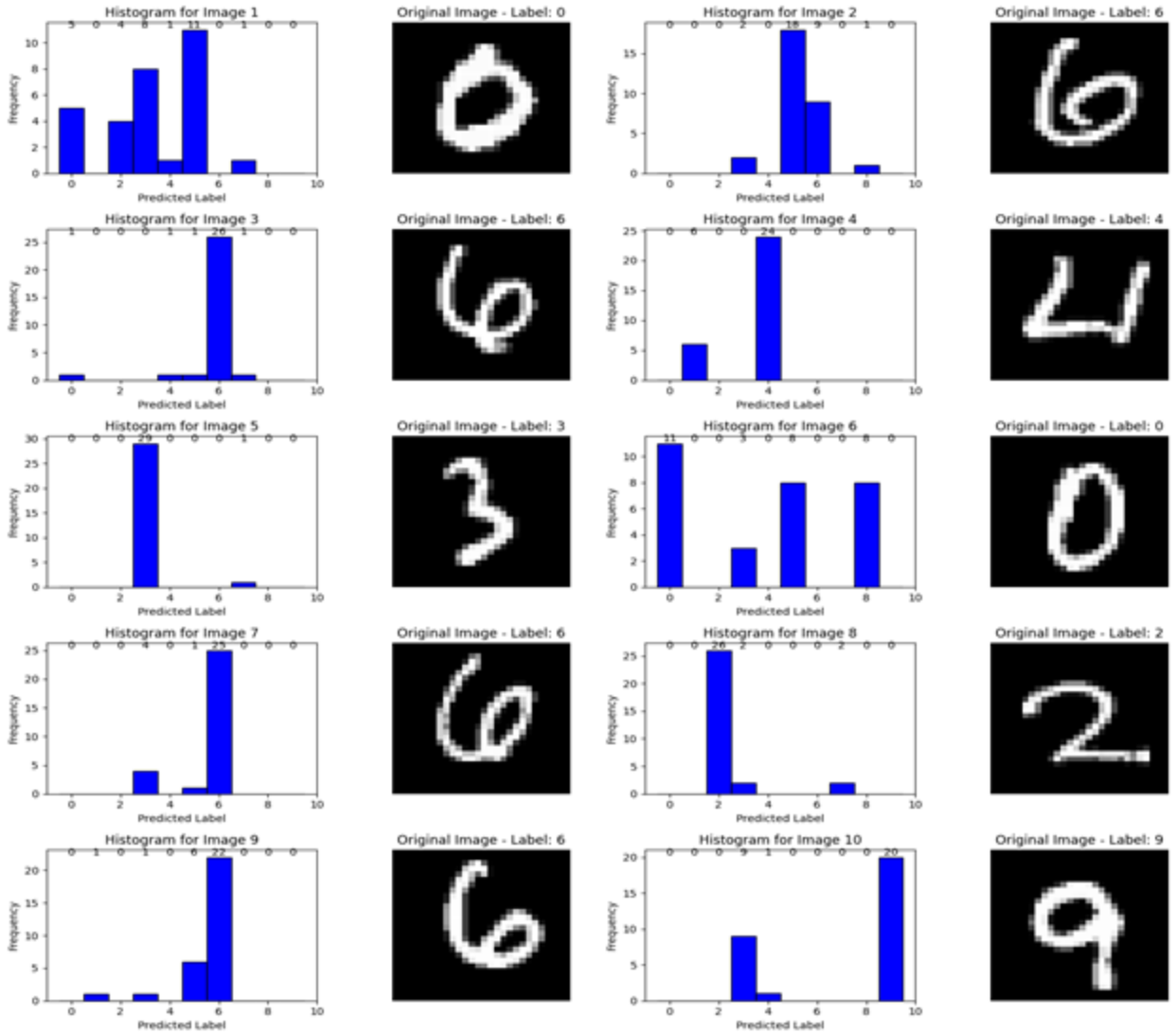
The accuracy, averaged over all predictions, increases with increase in drop rate.

Dropout On MNIST

- Dropout drops random pixels from the image, basically random pixels will turn black. We apply dropout 30 times to same image
- Thus, instead of showing the regular, “clean” images, only once to the trained model, we will show it the dropout images several times. We will then mode the predictions of each corresponding image and take that as our final guess.
- **dropout = nn.Dropout(p=0.3)**






Histogram for Image 1
Original Image - Label: 0
Histogram for Image 2
Original Image - Label: 6
Histogram for Image 3
Original Image - Label: 6
Histogram for Image 4
Original Image - Label: 4
Histogram for Image 5
Original Image - Label: 3
Histogram for Image 6
Original Image - Label: 0
Histogram for Image 7
Original Image - Label: 6
Histogram for Image 8
Original Image - Label: 2
Histogram for Image 9
Original Image - Label: 6
Histogram for Image 10
Original Image - Label: 9
Frequency
Predicted Label

Dropout

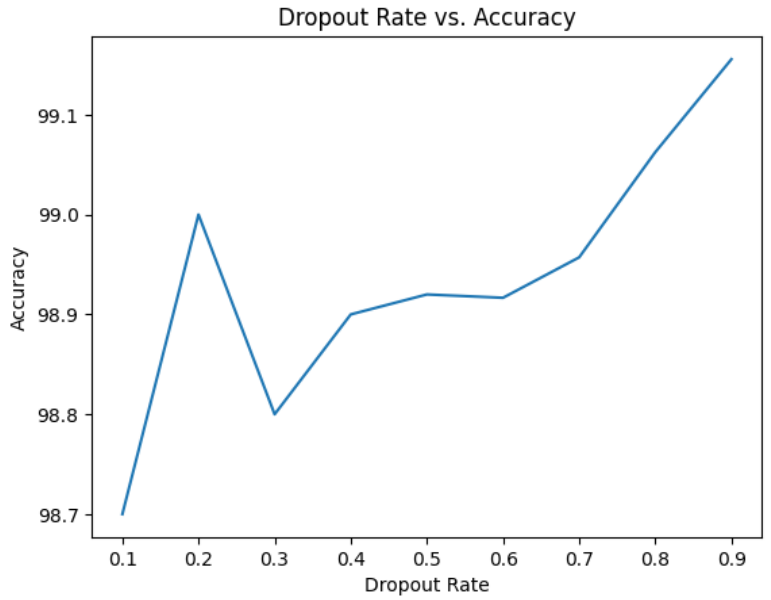
Dropout-Entropy

- Calculated Entropy as before only.
- We have predictions for 30 images after applying dropout
- Applied softmax on them and got probabilities which then we used to calculate entropy.
- **entropy = -np.sum(probas * np.log2(probas + 1e-10))**

```python
def calculate_softmax(predicted_labels):
    # Calculate e^x for each predicted label
    exp_values = np.exp(predicted_labels)

    # Calculate the sum of e^x values
    sum_exp_values = np.sum(exp_values)

    # Calculate softmax probabilities
    softmax_probs = exp_values / sum_exp_values

    return softmax_probs
```

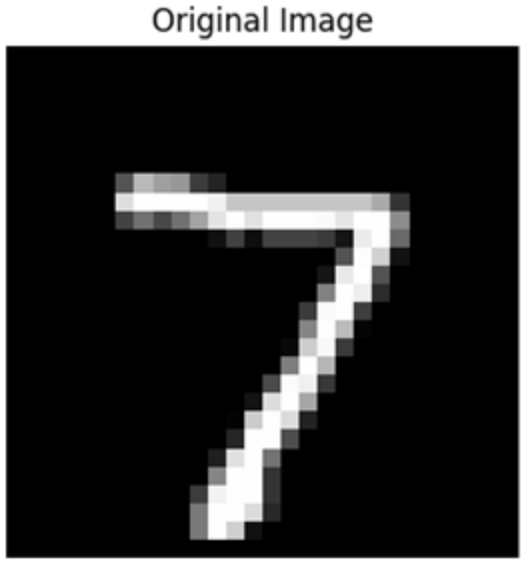
Drop Rate Vs Accuracy Curve

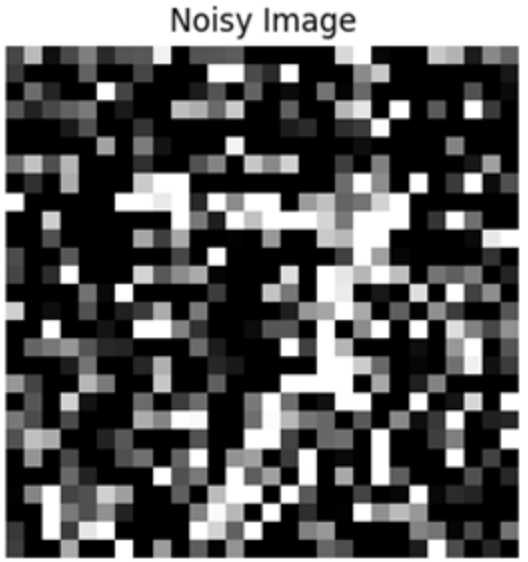
- We got entropy values, sorted them in descending order and dropped samples with drop rate 10%, to see the effect of dropping samples with higher uncertainty on accuracy



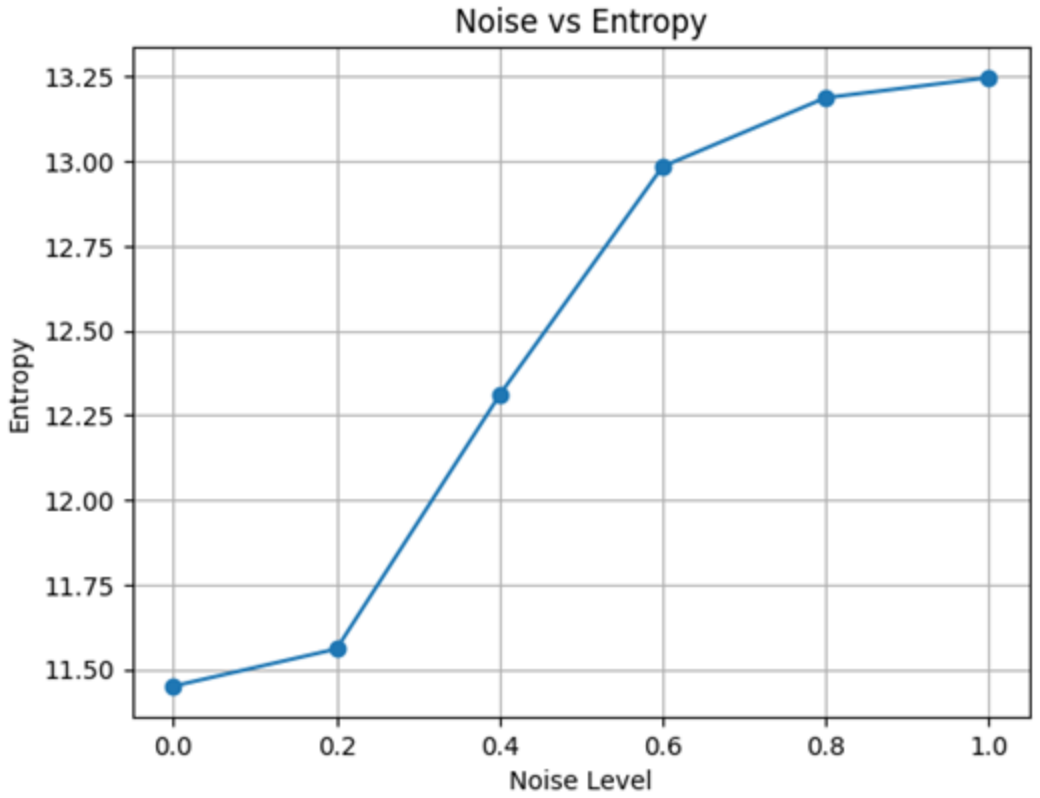
Entropy Vs Noise Curve

- We added Gaussian noise to the images with different noise levels and plot Entropy vs noise curve
- Here for entropy, we got predictions over whole test set and then applied softmax on those predictions and thus got one entropy value for one noise level
- Gaussian noise with standard deviation = 0.7 and mean 0





Entropy Vs Noise Curve

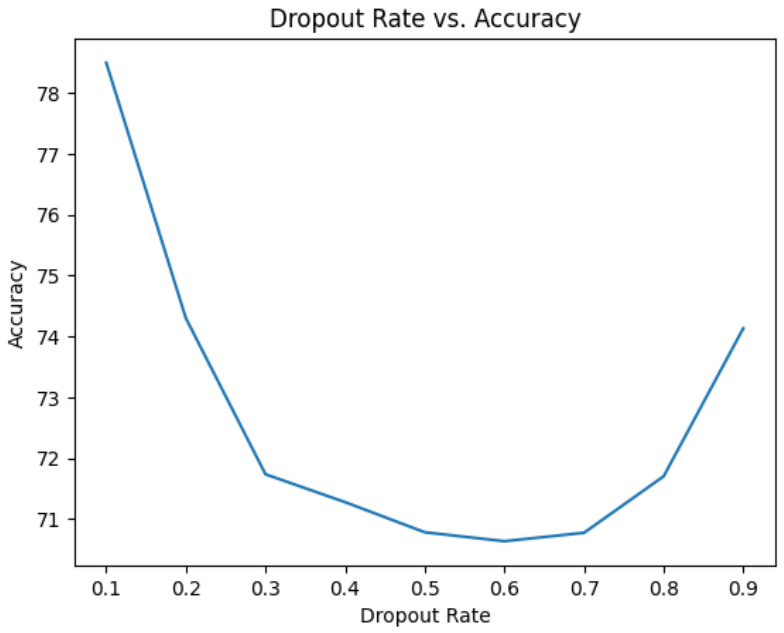


Dropout On CIFAR-10

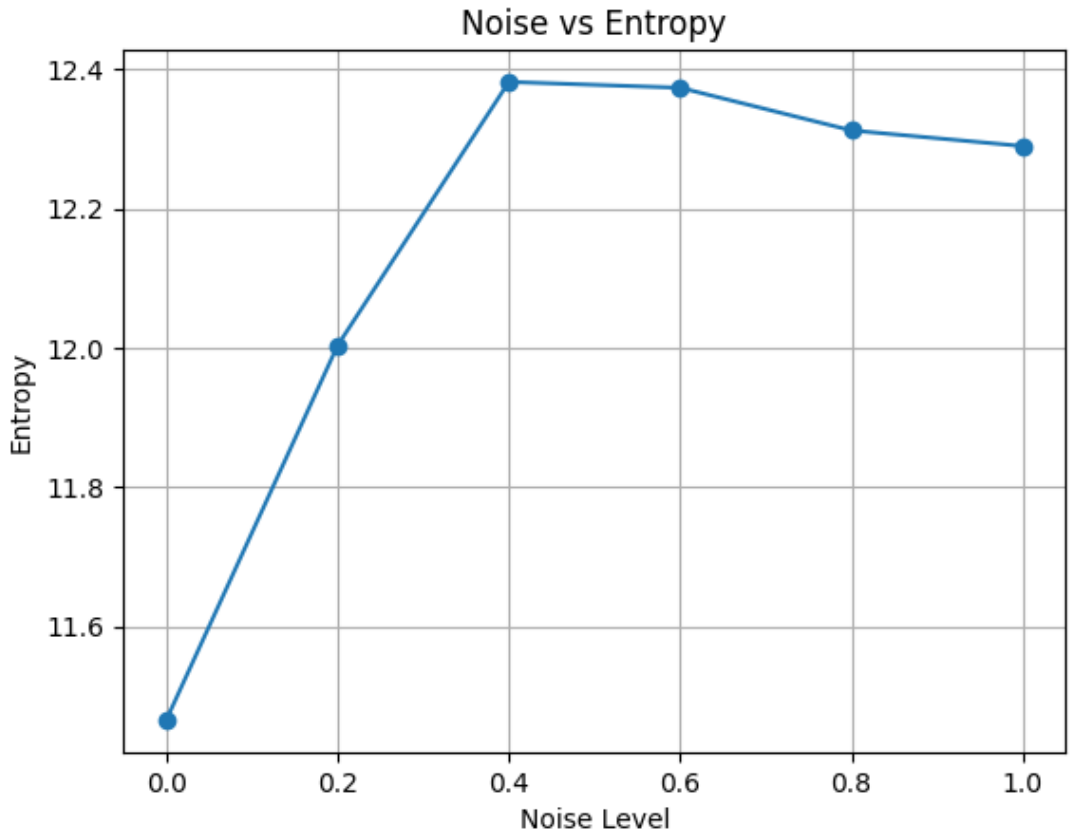
- Done Similar plots for CIFAR10

Drop Rate Vs Accuracy Curve

- We got entropy values, sorted them in descending order and dropped samples with drop rate 10%, to see the effect of dropping samples with higher uncertainty on accuracy



Entropy Vs Noise Curve



Utility

In this project, we have created a benchmark function that takes a dataset and a trained model as input and outputs the uncertainty associated with a prediction along with the prediction itself in classification problems.

Two functions are available for uncertainty quantification- one uses TTA while the other uses test time dropout.

Learnings

- This project has provided us with a deeper understanding of the different **sources of uncertainty** in machine learning and the various **methods used to measure it**.
- Additionally, we were introduced to the concept of **test time augmentation** and **test time dropout** and learned how it can enhance the robustness of predictions.
- We also learnt about **entropy** in machine learning and the behaviour of entropy curves in different settings.

Challenges Faced

- **Lack of consolidated literature**

All the literature available on uncertainty estimation is fragmented across different papers in which a UQ method only being used for a specific problem.

There are only a few papers which generalize and compare different UQ methods.

Thus, we had to read a lot of papers to fully understand the topic.

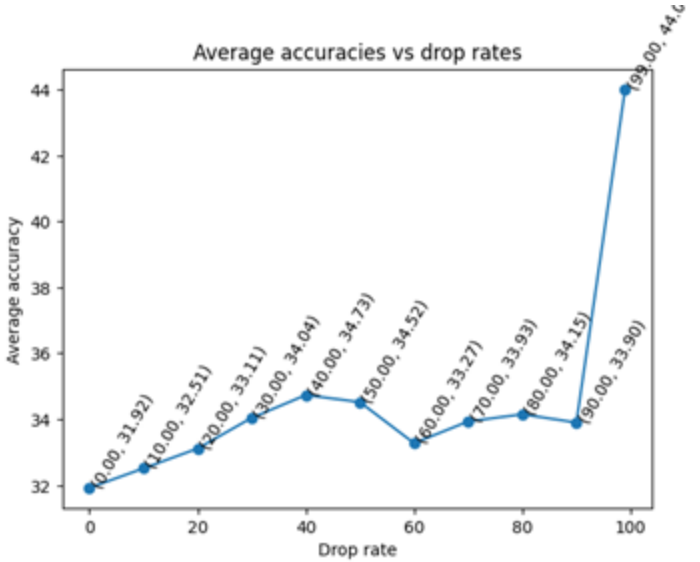
Challenges Faced

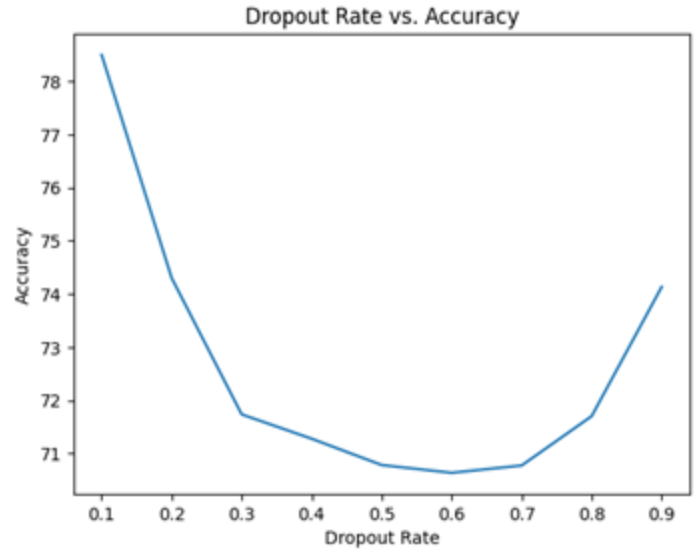
- **Choice of transformations**

Some transformations like Horizontal flip etc. were giving issues with the MNIST dataset and these transforms change the digits.

Comparison of results

For CIFAR10 dataset, TTA works but dropout doesn't work properly, as we can see from the drop rate vs accuracy curve.





Future Work

- **OOD Prediction:** The benchmark UQ prediction can be used to find OOD samples by thresholding on entropy values.

Refernces

1. Gruber, C., Schenk, P. O., Schierholz, M., Kreuter, F., & Kauermann, G. (2023). **Sources of Uncertainty in Machine Learning--A Statisticians' View.** arXiv preprint arXiv:2305.16703.
2. Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., ... & Nahavandi, S. (2021). **A review of uncertainty quantification in deep learning: Techniques, applications and challenges.** Information fusion, 76, 243-297.
3. Sensoy, M., Kaplan, L., & Kandemir, M. (2018). **Evidential deep learning to quantify classification uncertainty.** Advances in neural information processing systems, 31.
4. Jospin, L. V., Laga, H., Boussaid, F., Buntine, W., & Bennamoun, M. (2022). **Hands-on Bayesian neural networks—A tutorial for deep learning users.** IEEE Computational Intelligence Magazine, 17(2), 29-48.
5. Y. Gal ,& Z. Ghahramani(2016). **Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning.** arXiv:1506.02142