Skin Cancer Detection

A deep probabilistic approach

PYTORCH

ROADMAP

Motivation & Introduction

What we did and why we did it

Dataset

What we have to deal with

CNN & Methods

Which NN, why and how?

The Bayeasian prospective

Distributions and uncertainty

Experiments

We are doing science

Results & Conclusion

What we did right, what we did wrong.
What we would have liked to do if we had time

MOTIVATION

1) **Skin cancer** statistic increasing every year

2) Melanoma is its deadliest form

3) 5-year **survival rate**:
   97% if detected in its earliest stages
   15% if detected in its latest

4) Most effective solution is **timely diagnosis**

MOTIVATION:

Skin cancer diagnostic procedure

Visual inspection by dermatologists	Dermoscopy imaging	Reliable automated method for the classsification of dermoscopy skin cancer images
80% accuracy: experience > 10 years 61% accuracy: experience 3-5 years lower accuracy for less experienced dermatologists [1]	Due to skin lesion similarity visual accuracy does not increase very much	DNN: > 90% accuracy CNN (ResNet125-Densnet121): 85% accuracy

[1] Morton CA, Mackie RM. Clinical accuracy of the diagnosis of cutaneous malignant melanoma. Br. J. Dermatol. 2017; 138(2):283–7

MOTIVATION

Our Goal

Deep NN are state of the art in many fields and achieved great results in skin-lesion classification.

However, they present some **limitations** (e.g. lots of data needed, not transparent, sensitive to attacks.)

In this project, instead of focusing on maximizing the **accuracy**, we will try to enlight their weaknesses and possibly, propose a solution.

The QUESTION

WOULD YOU PUT A STATE OF THE ART NEURAL NETWORK IN IN PRODUCTION EVEN IN LIFE-DEATH SITUATIONS?

WHAT WE WILL DO?

Try to investigate and incorporate **uncertainty** in prediction…

> We think that any diagnostic algorithm should be able to flag cases resulting in high uncertain prediction and requires more investigation by medical experts

HOW DO WE DO THAT?

… using a **Bayesian neural network.**

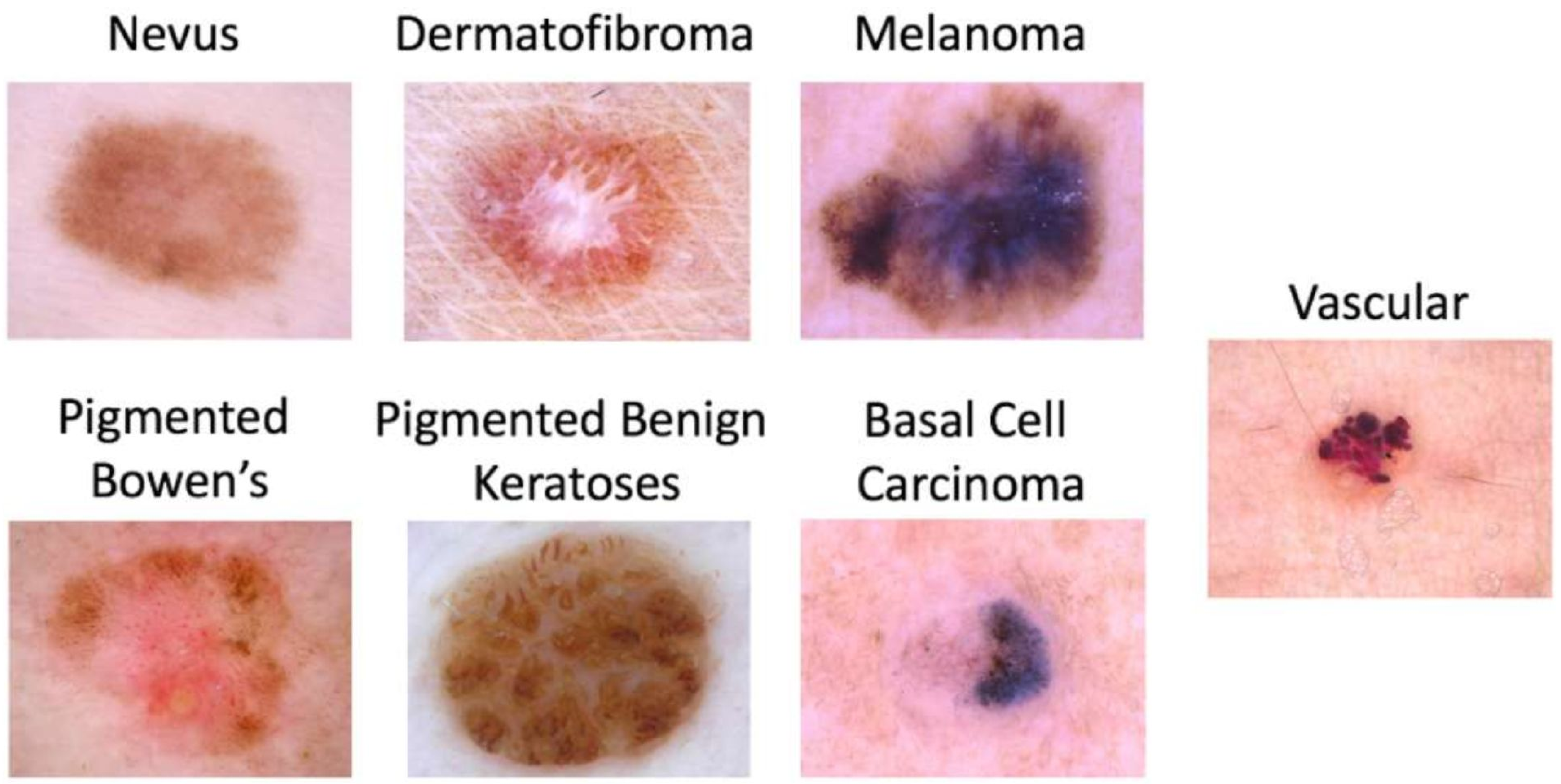
The DATASET

~ 10000 64 x 64 RGB dermatoscopic images from HAM10000 dataset

80% used for training

20% used for testing

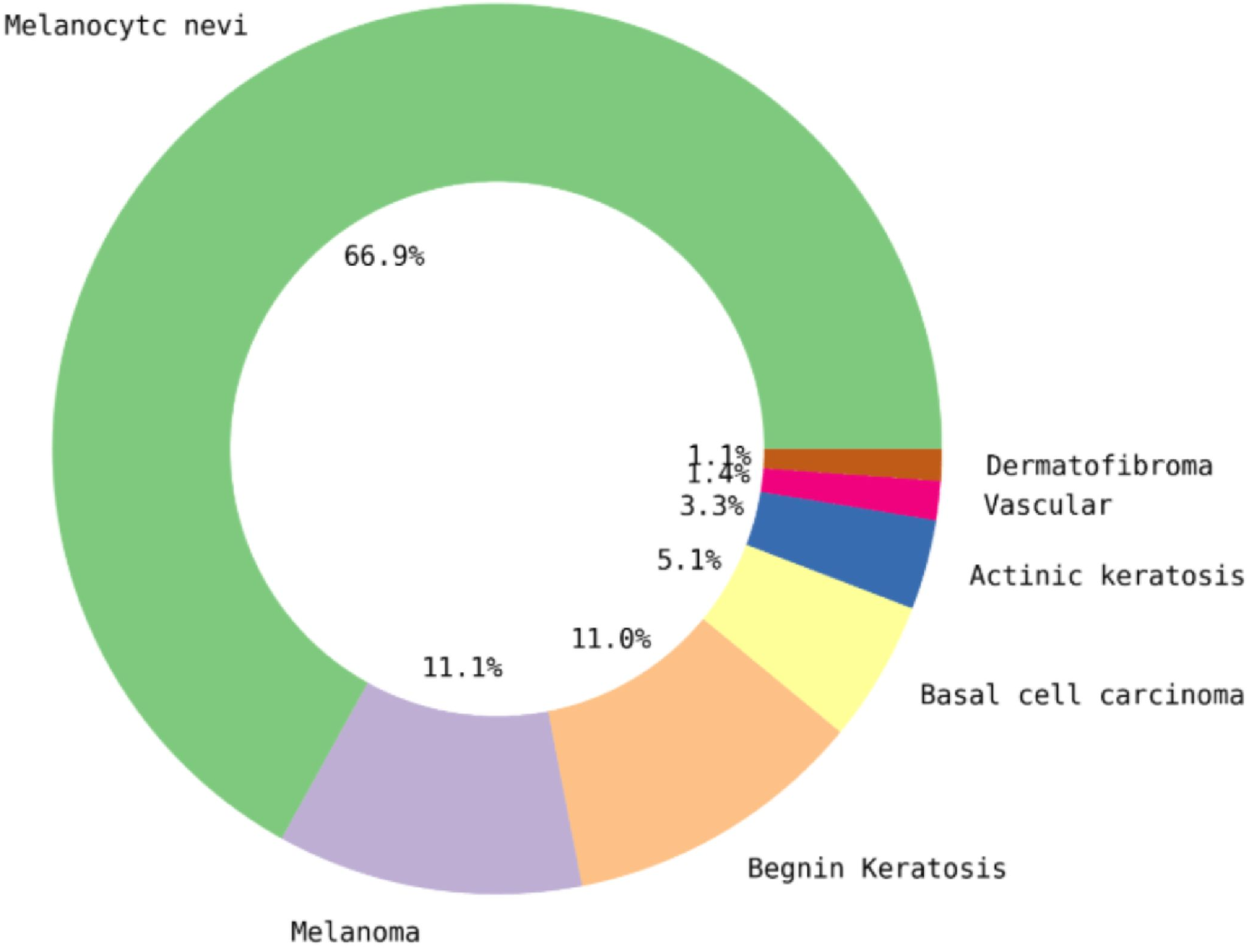
Problem: supervised multi-class classification



DATASET

Issues

- Very low pixel resolution
- Unbalance distribution of images among seven classes
- Not clear distinction between classes



DATASET

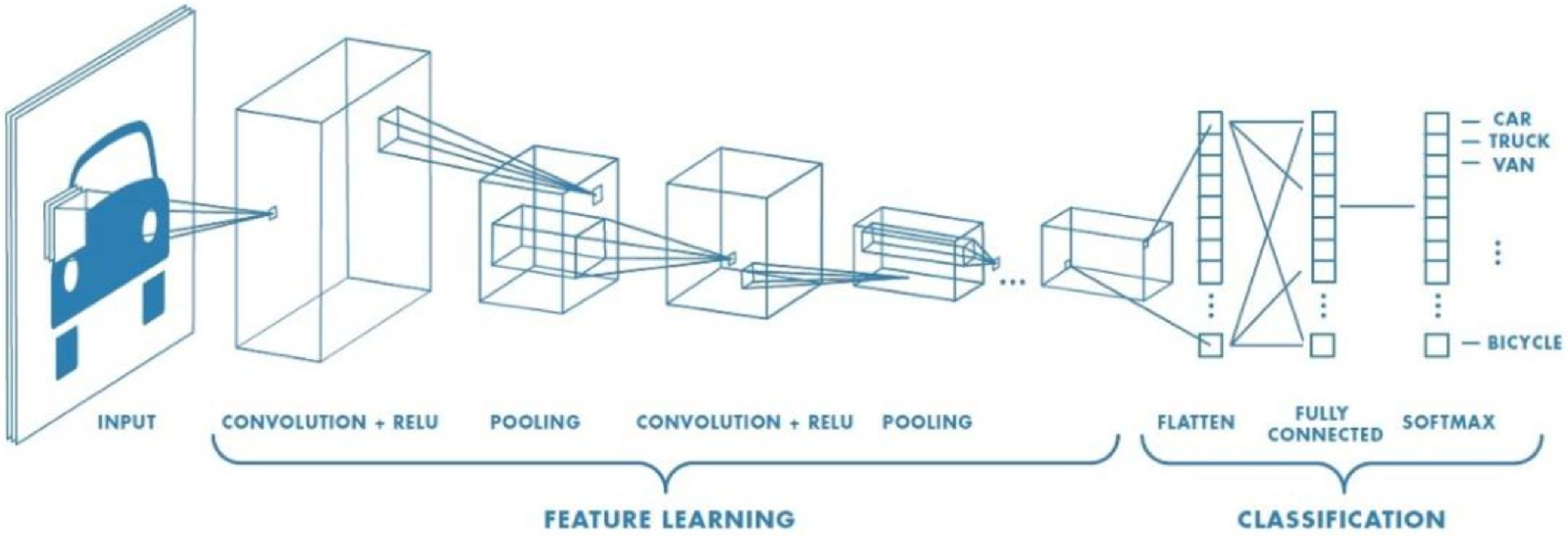
Balancing dataset: Data augmentation

- We conducted data augmentation of minority classes in the dataset
    - Modify **saturation**, **contrast**, and **brightness** by random factors sampled from an uniform distribution of [0.7, 1.3]
    - **Random crop** of size 0.4–1.0 of the original area, and 3/4 – 4/3 of the original aspect ratio
    - **Rotation** by up 90°
- Normalization

Packages: "imbalanced-learn" and "torchvision"

Convolutional Neural Network

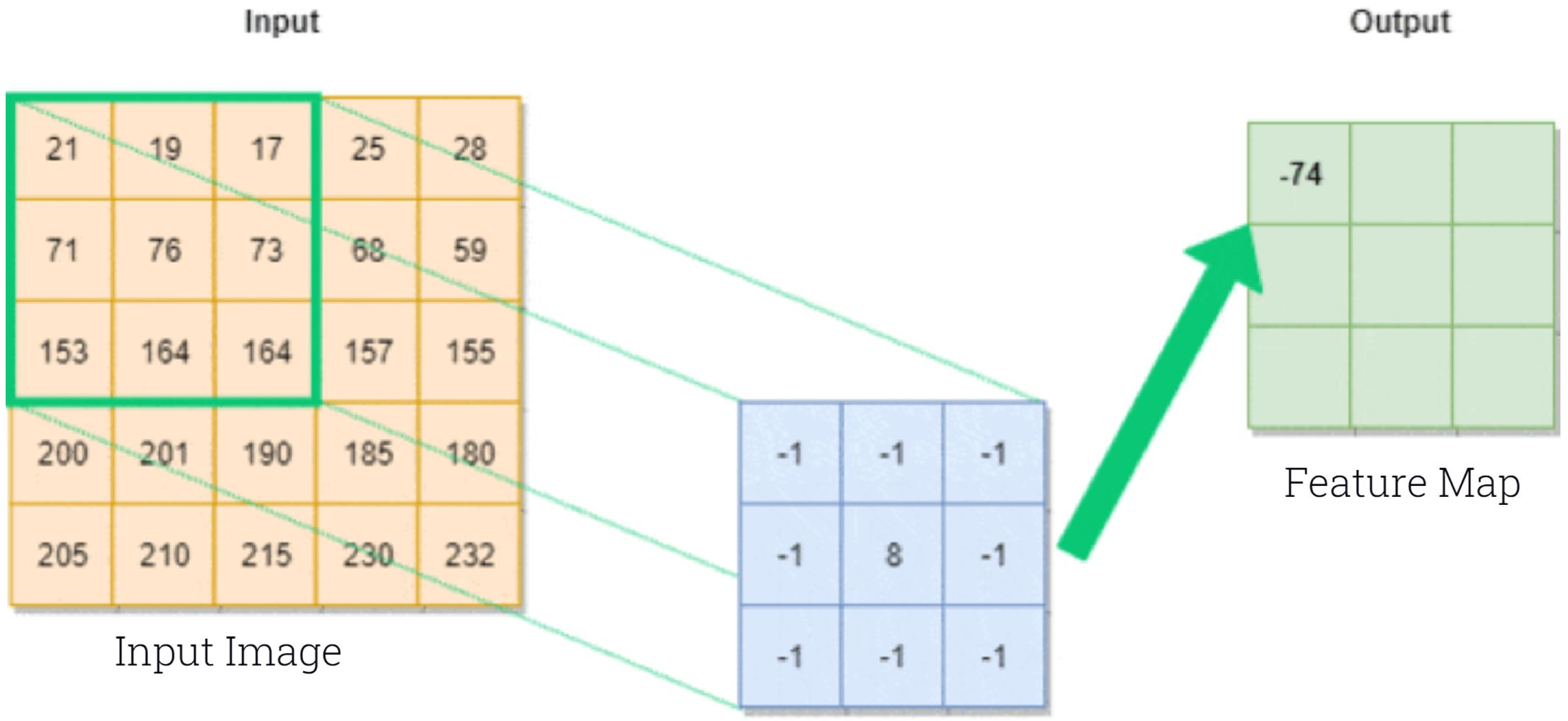
- Promising results in biomedical imaging classification according to literature



Convolutional Neural Network

Convolution Layer

- Extract features from a small square of the input image preserving the spatial relationship between pixel
- Local connectivity: each neuron is connected only to a subset of the input image



Input Image

Filter or Kernel

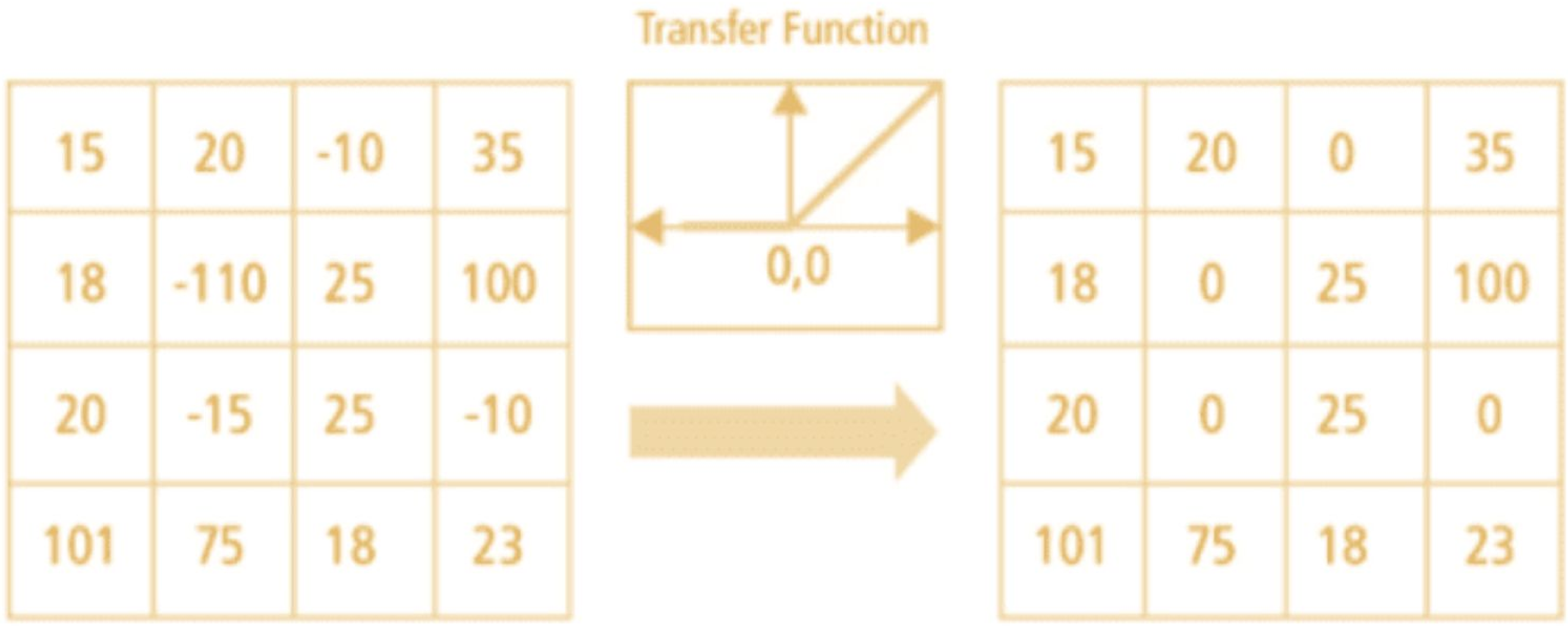
Feature Map

Convolutional Neural Network

Detector Stage and Pooling

- Detector Stage

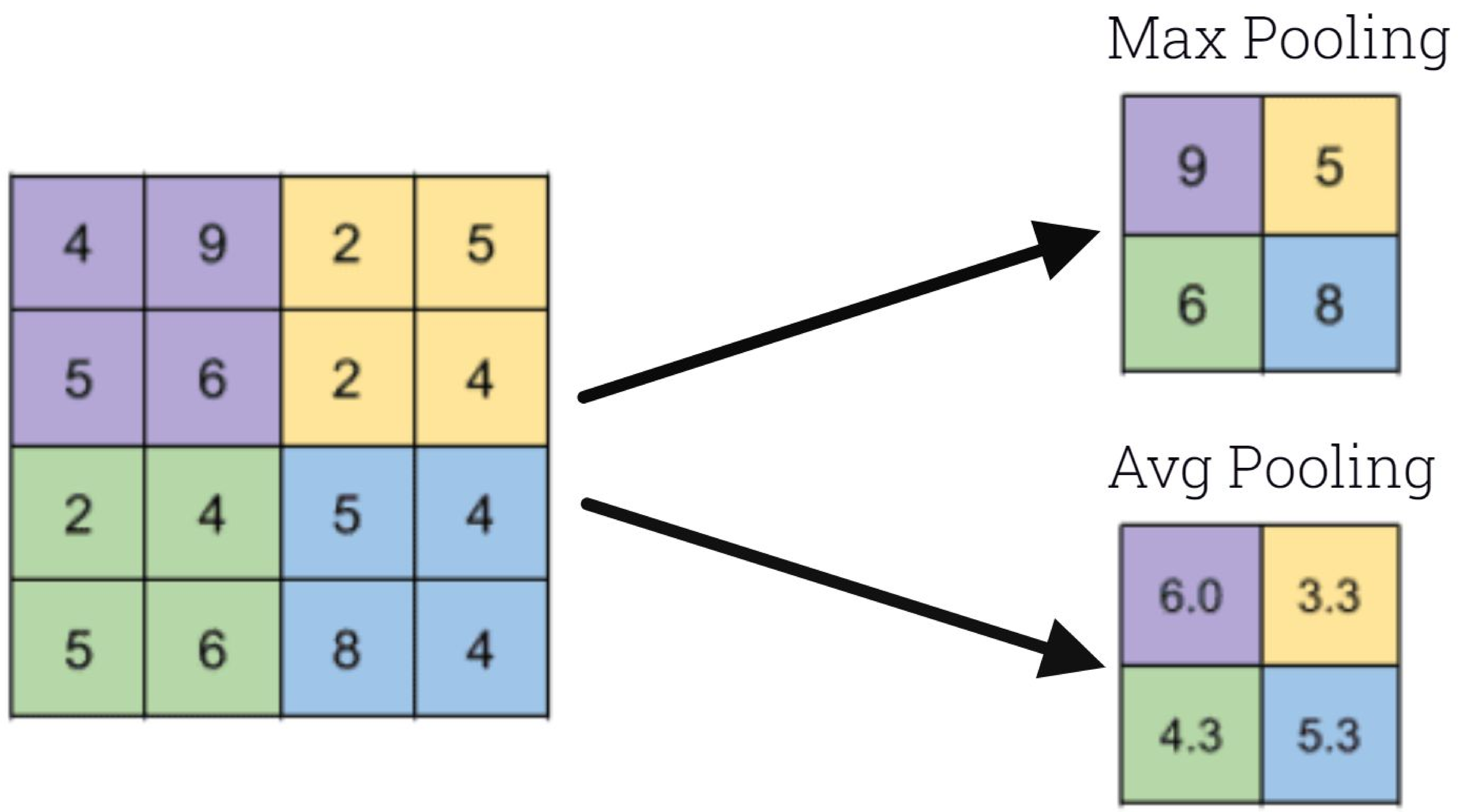
  Introduce **non linearity** using an activation function (most used ReLu)



- Pooling Layer

  Reduce the size of the input image applying a downsampling function and speed up computation

  Retain most important feature enforcing **invariance to small translation**



Convolutional Neural Network

Classification layer

- **Flatten layer**

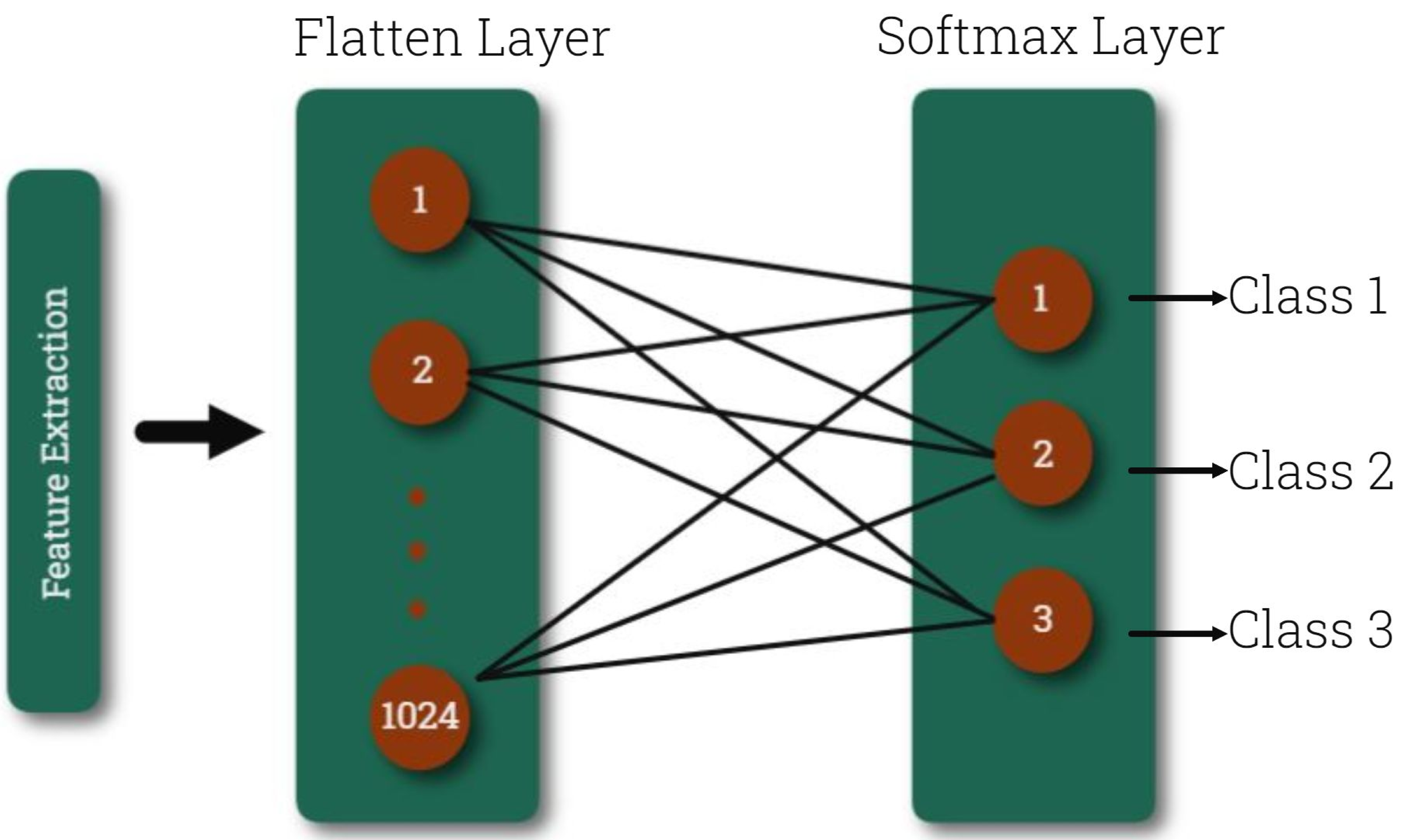
  CNN require 1-D feature vector

- **Fully conected layer**

  Combine all the previous local features in order to perform classification

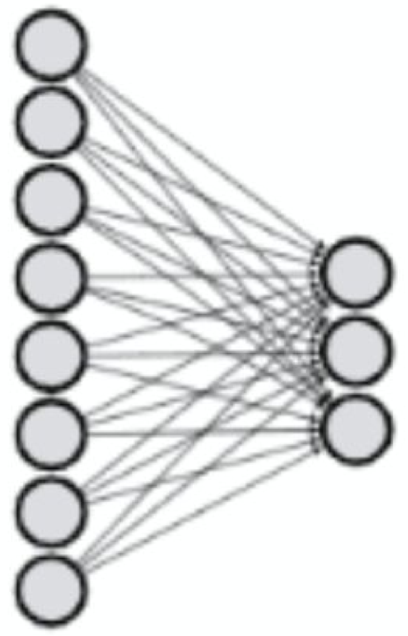
- **Softmax** activation function

  Most used in multi-class classification
  Gives the discrete probability distribution over all the classes

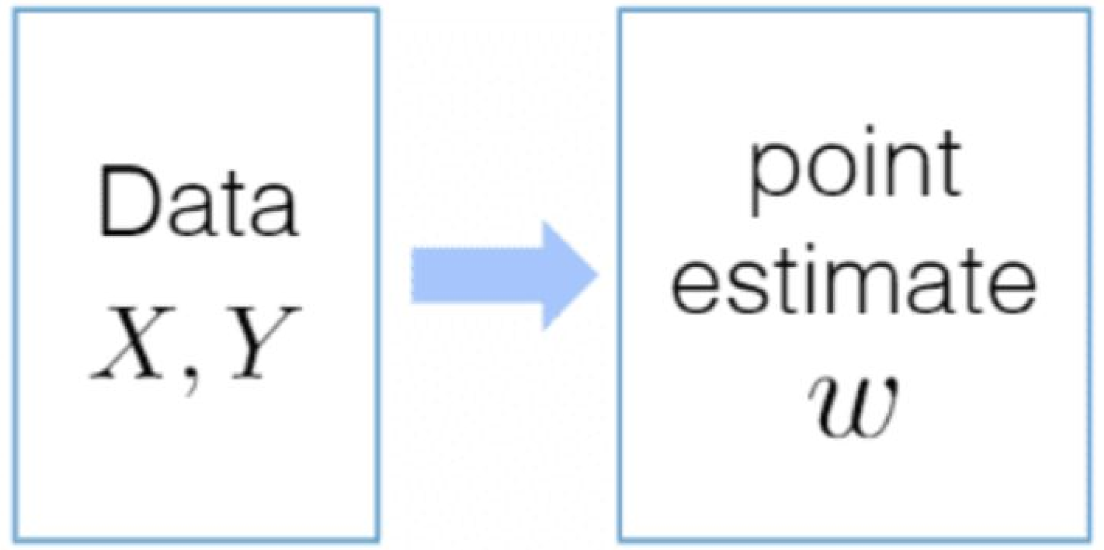


Deterministic Neural Networks

- Deterministic weights and bias



- Training

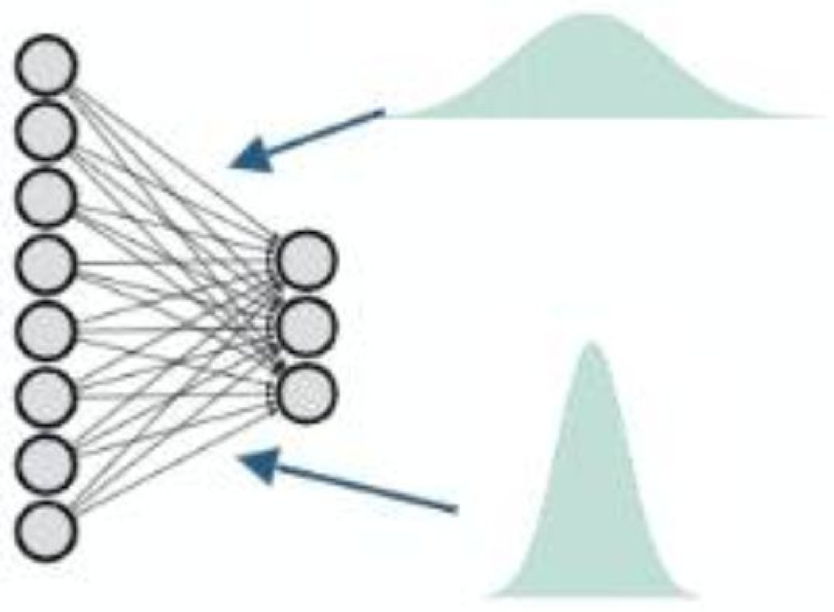


- Prediction

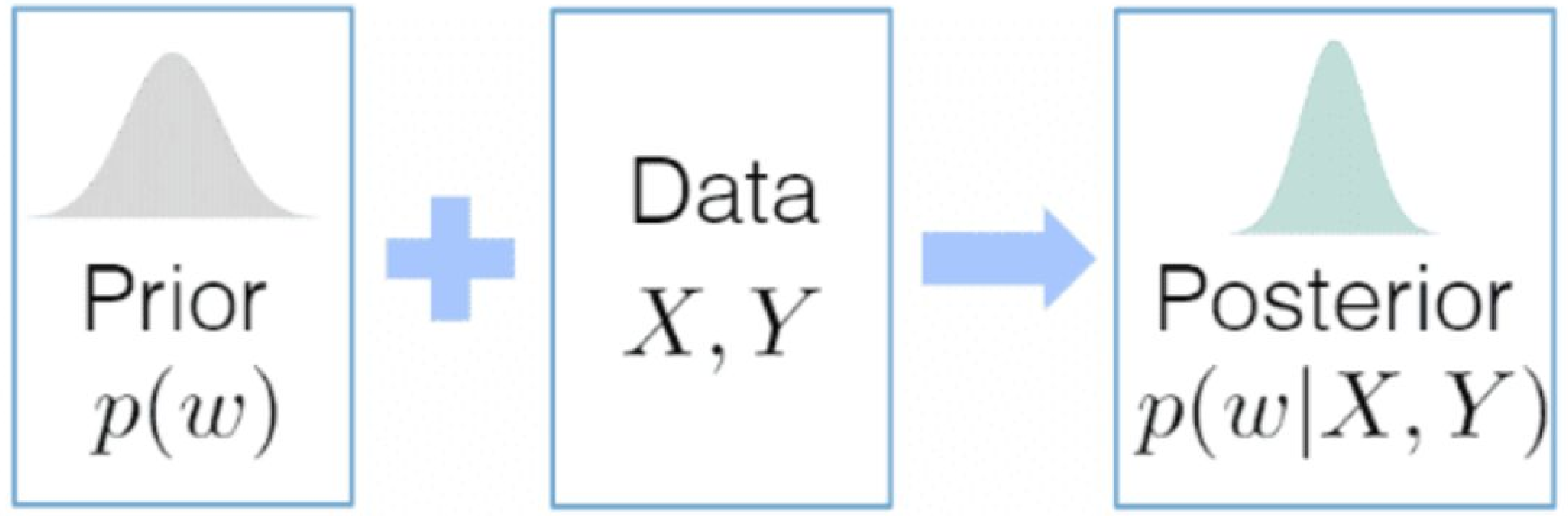
$$p(y_*|x_*, w)$$

Bayesian Neural Network

- Stochastic weights and bias



- Training



- Prediction

$$\mathbb{E}_{p(w|X,Y)}p(y_*|x_*, w)$$

Why go Bayesian?

- Neural Networks

  Point estimation of output (class with the highest probability)

  Overconfident & represents uncertainty poorly

  Critical decisions should be based on **robust uncertainty estimates** especially in biomedical field.

- Bayesian Neural Networks

  Distribution over the output

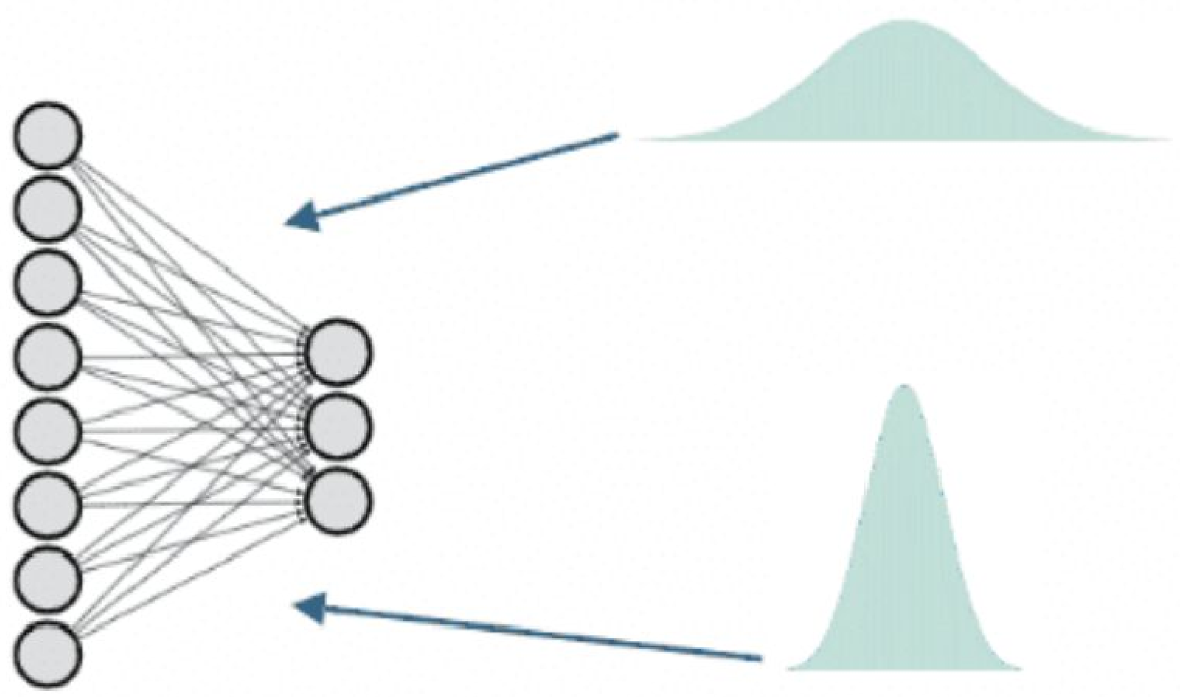
  Better calibration of uncertainty estimation

  Hard to choose the prior

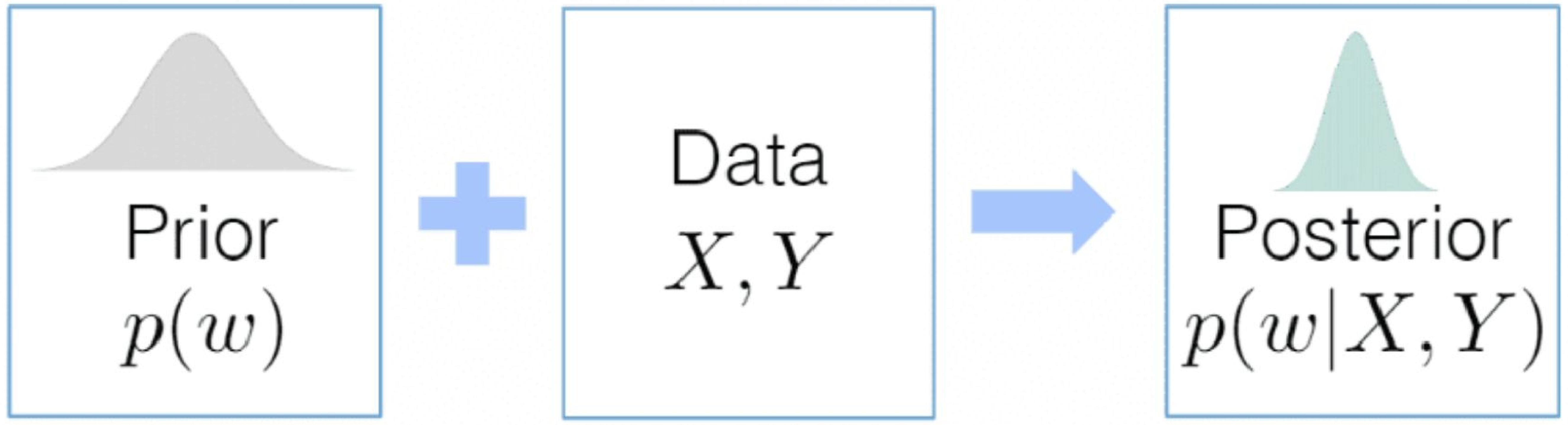
  Less prone to overfit data

Training Bayesian Neural Networks

Stochastic weights:



Bayesian Inference:

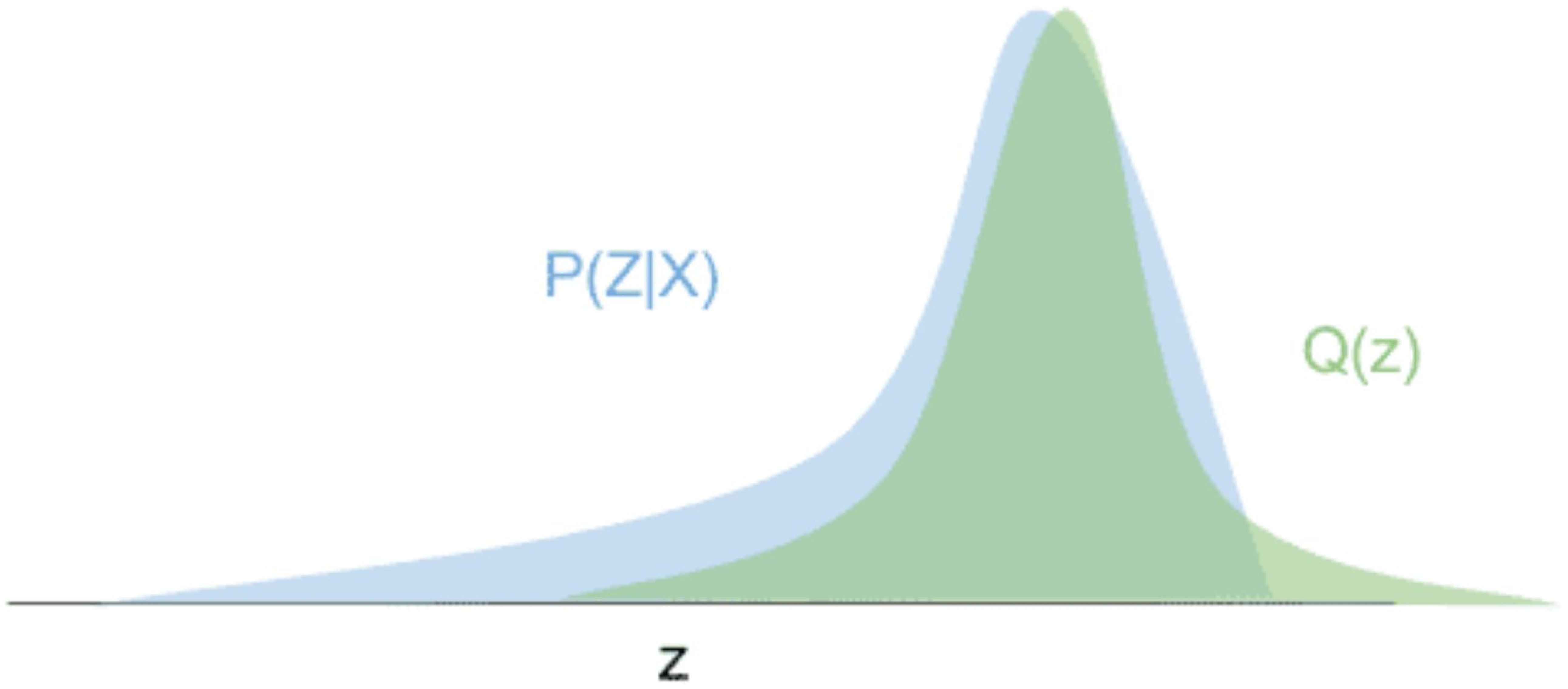


Posterior is intractable in Neural Network → Approximate it using **Variational Inference**

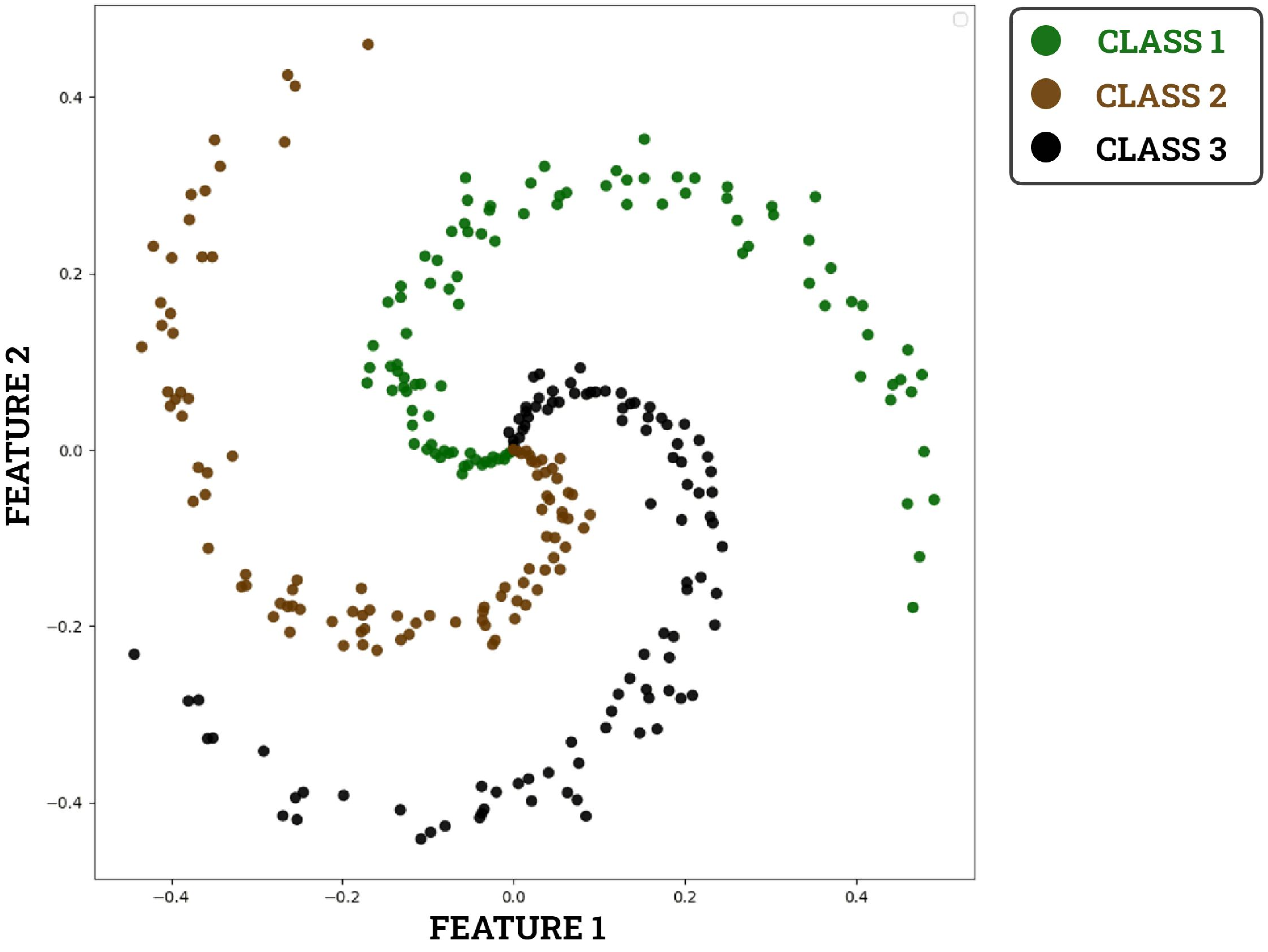
Training Bayesian Neural Networks

Variational Inference

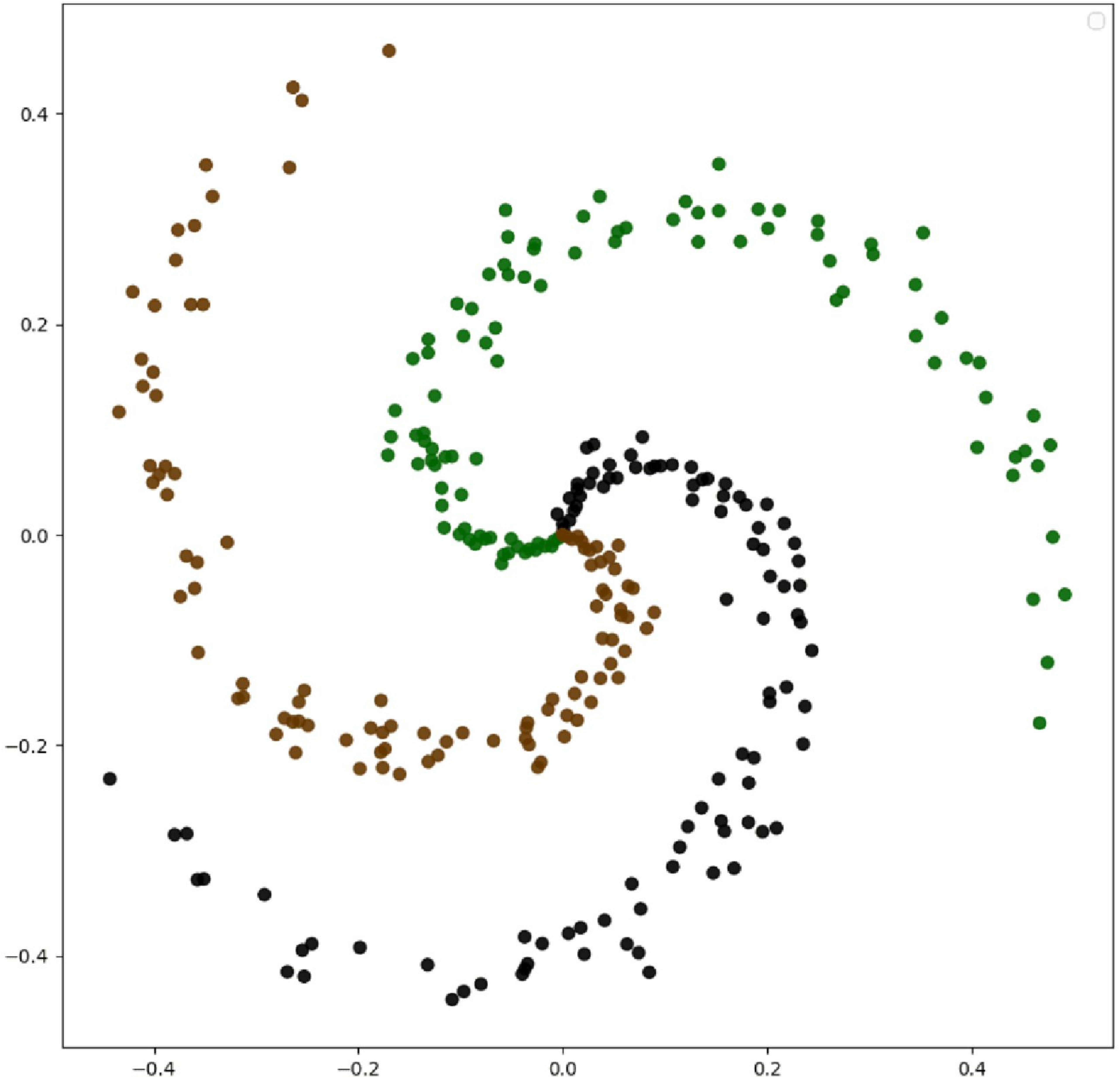
- Approximate true intractable distribution P(Z) with a simpler tractable one Q(z)
    - Kullback-Leibler (KL) divergence is a measure for similarity
    - KL = - ELBO + constant
    - Maximize ELBO using Gradient Descent



WARMUP:
A 2 FEATURES
EXAMPLE



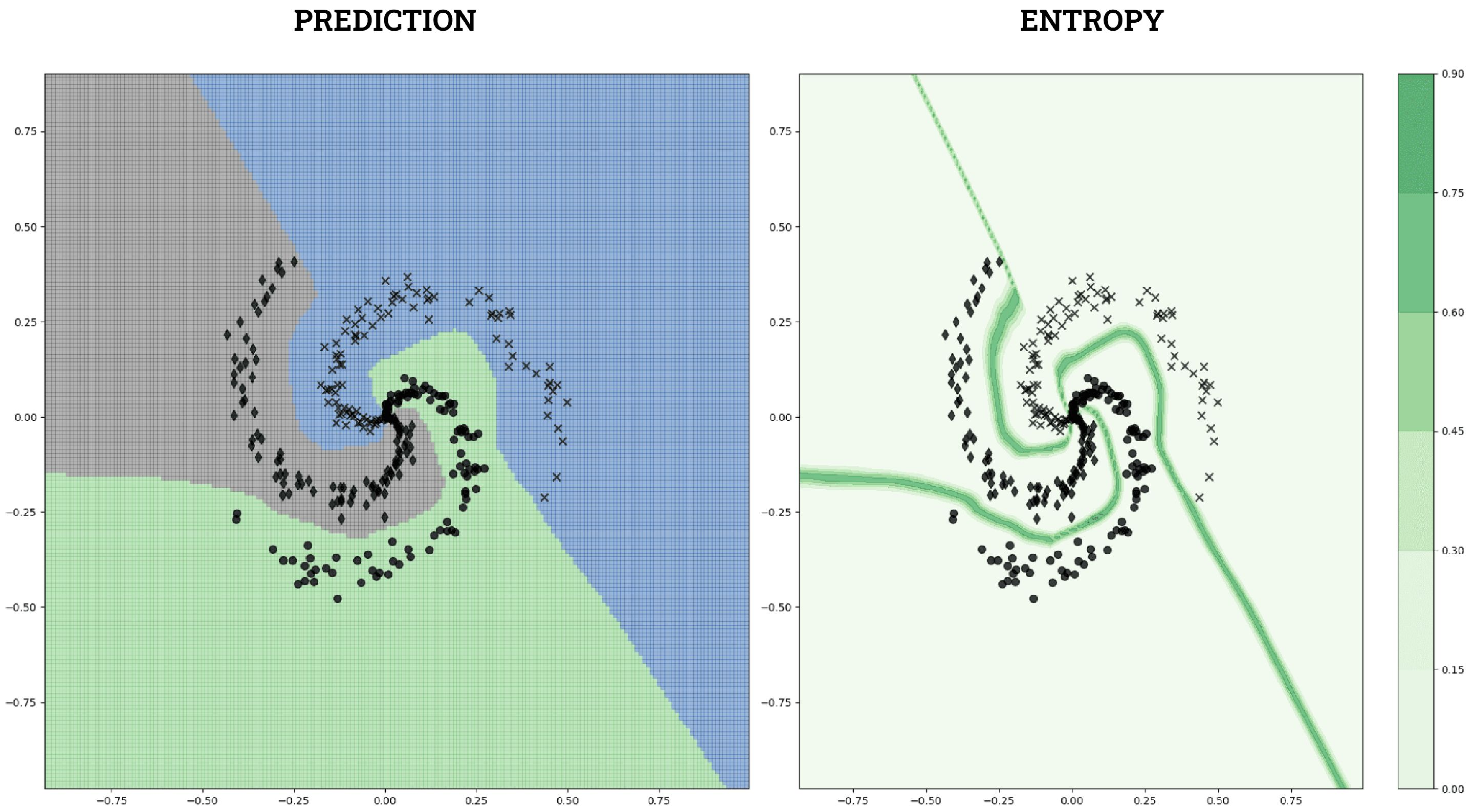
CLASS 1
CLASS 2
CLASS 3
FEATURE 2
FEATURE 1
0.4
0.2
0.0
−0.2
−0.4
−0.4
−0.2
0.0
0.2
0.4



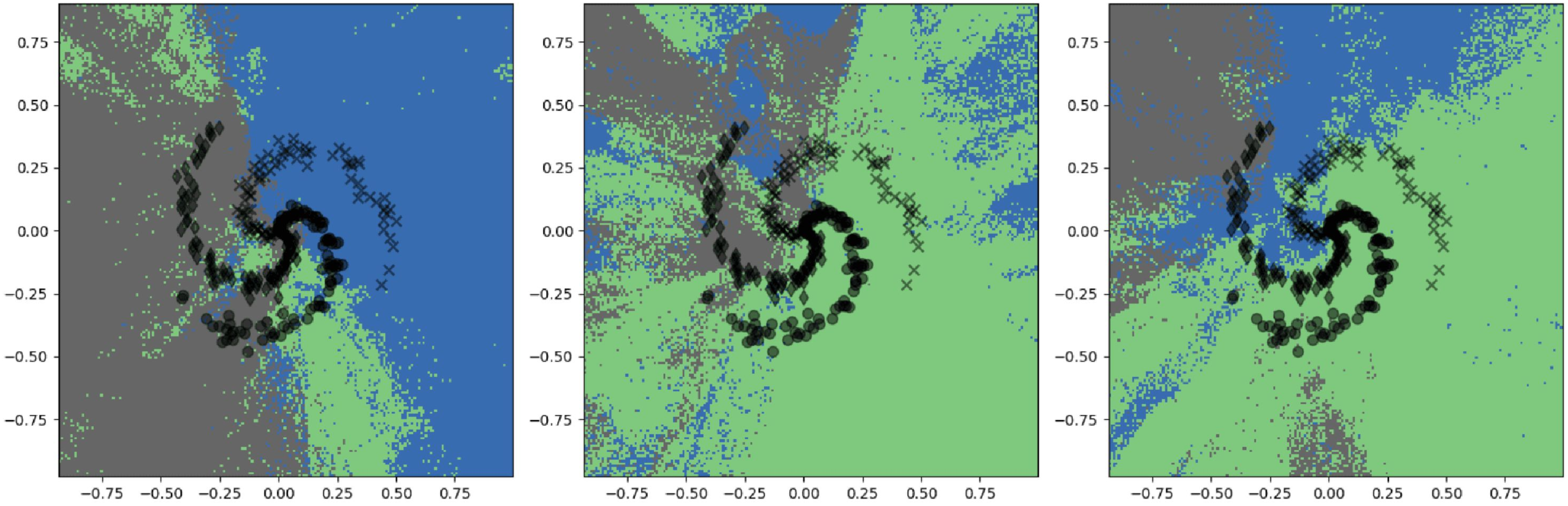

```
def __init__(self, num_hidden=10):
    super(NN, self).__init__()
    self.layer1 = torch.nn.Linear(2, num_hidden)
    self.layer2 = torch.nn.Linear(num_hidden, num_hidden)
    self.layer3 = torch.nn.Linear(num_hidden, 3)
    self.activation = torch.nn.ReLU()

def forward(self, x):
    z = self.activation(self.layer1(x))
    z = self.activation(self.layer2(z))
    return self.layer3(z)
```

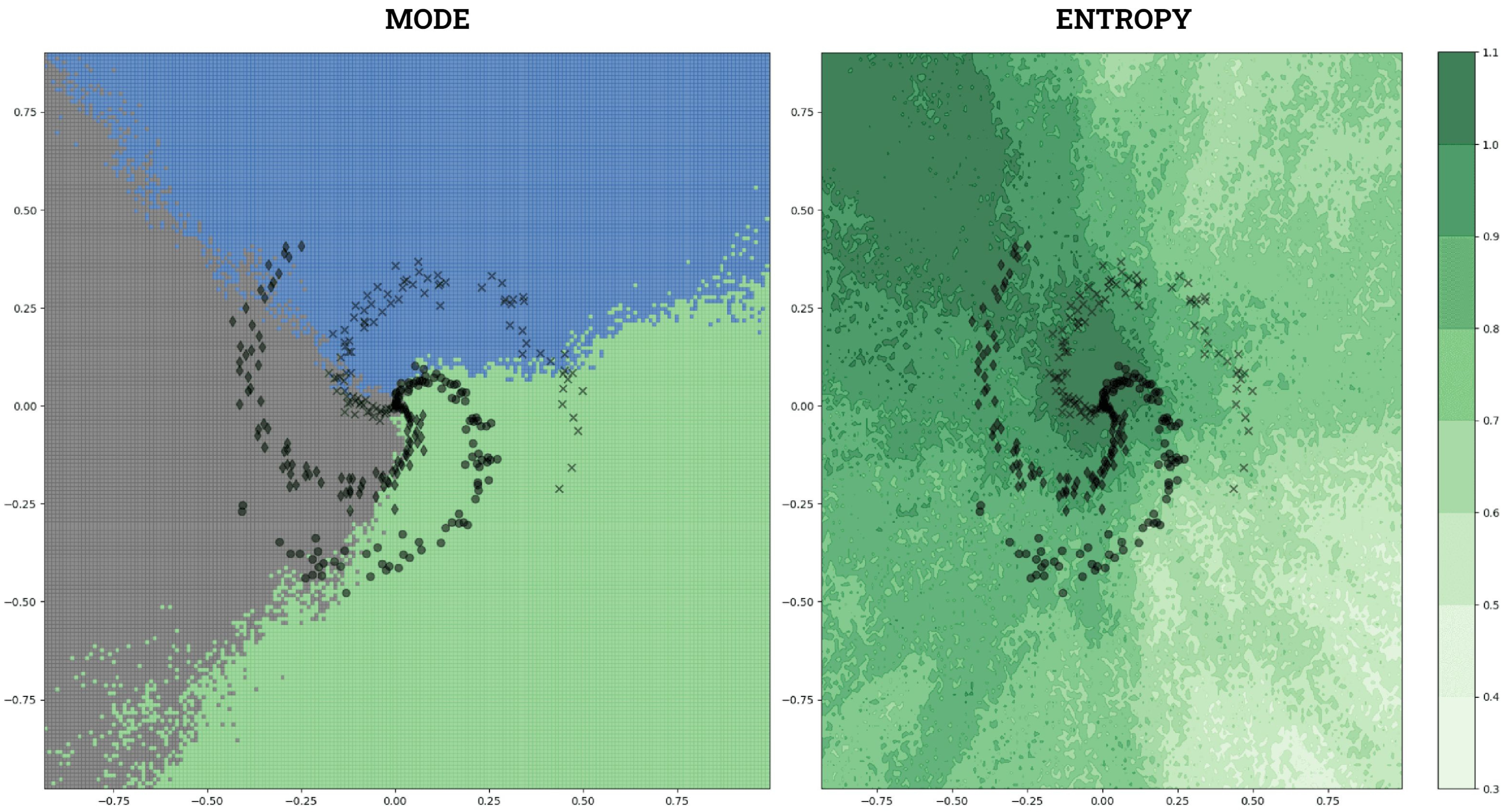
Final verdict of NN



3/100 BNN Predictions

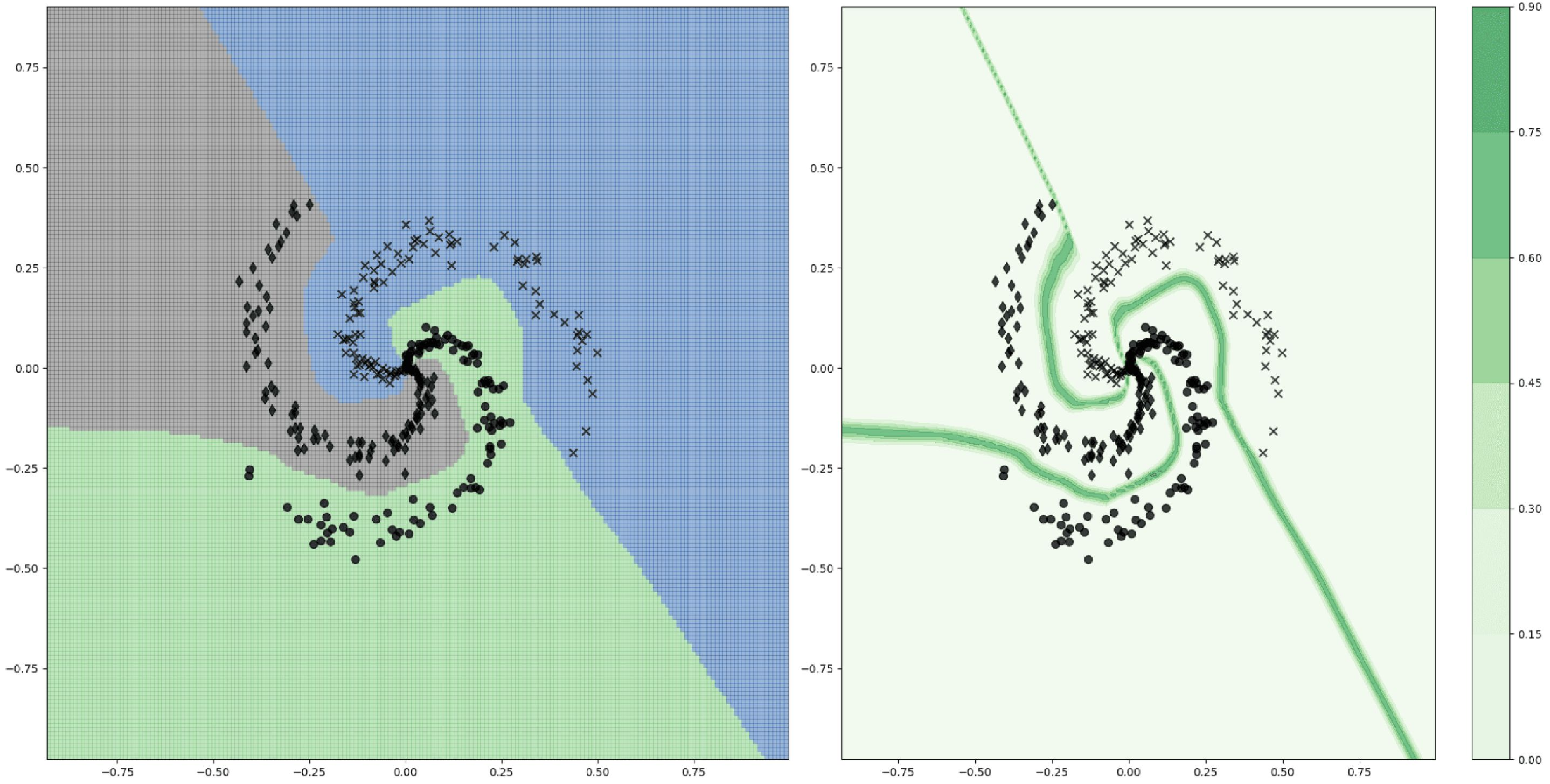


Final verdict of BNN



Final verdict of NN

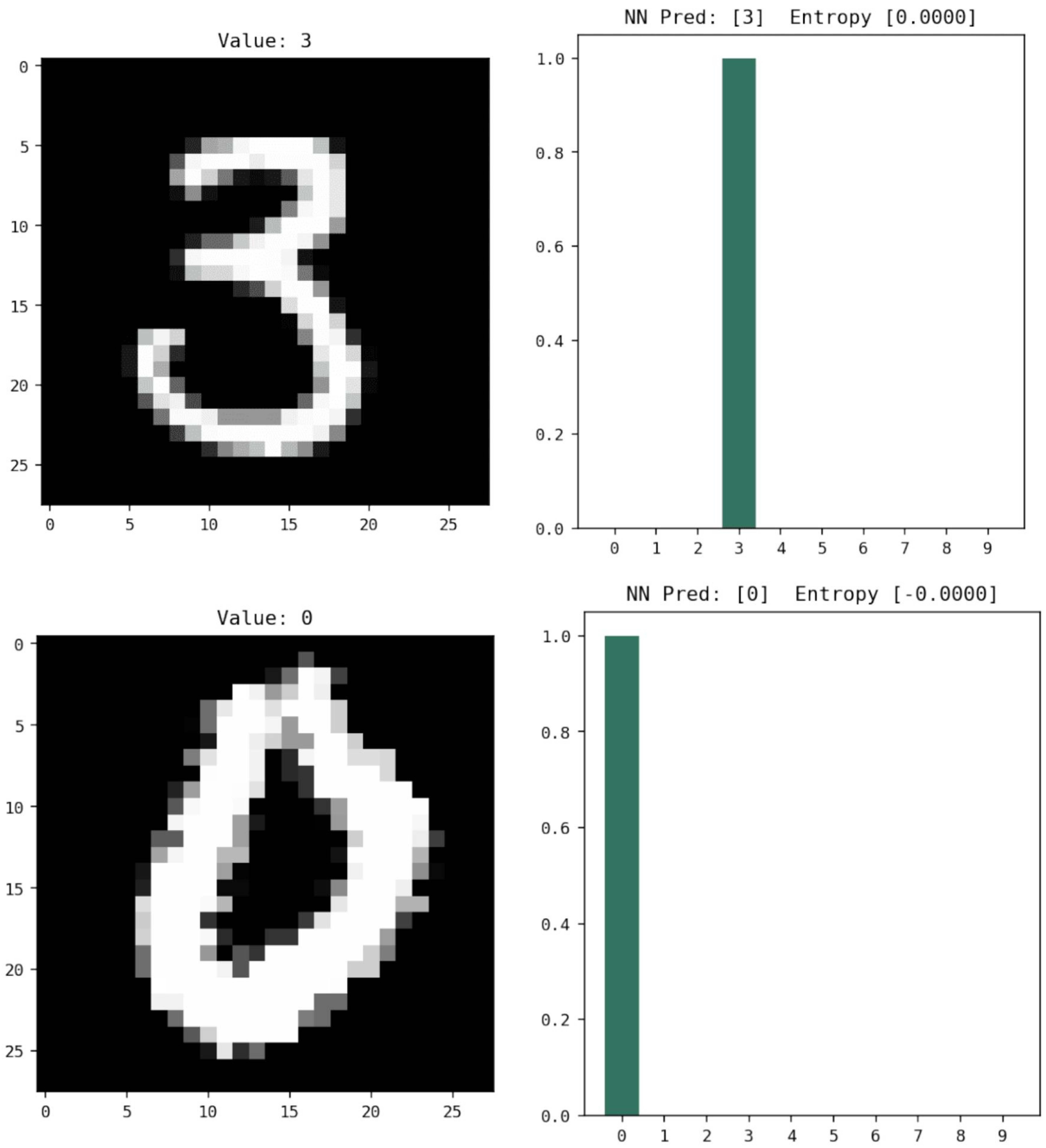
PREDICTION **ENTROPY**



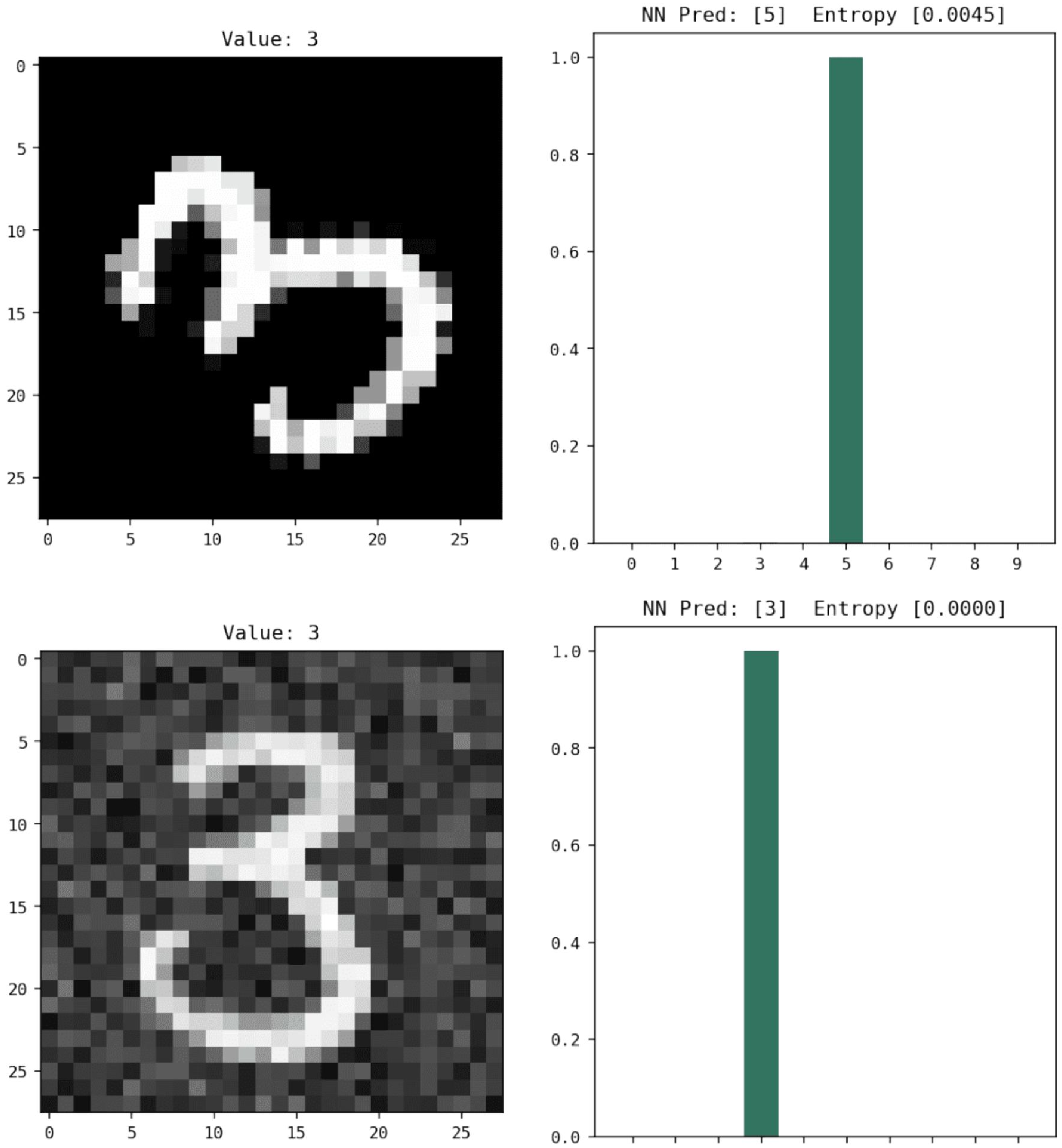
NO DEEP LEARNING PROJECT
IS COMPLETE WITHOUT:

MNIST

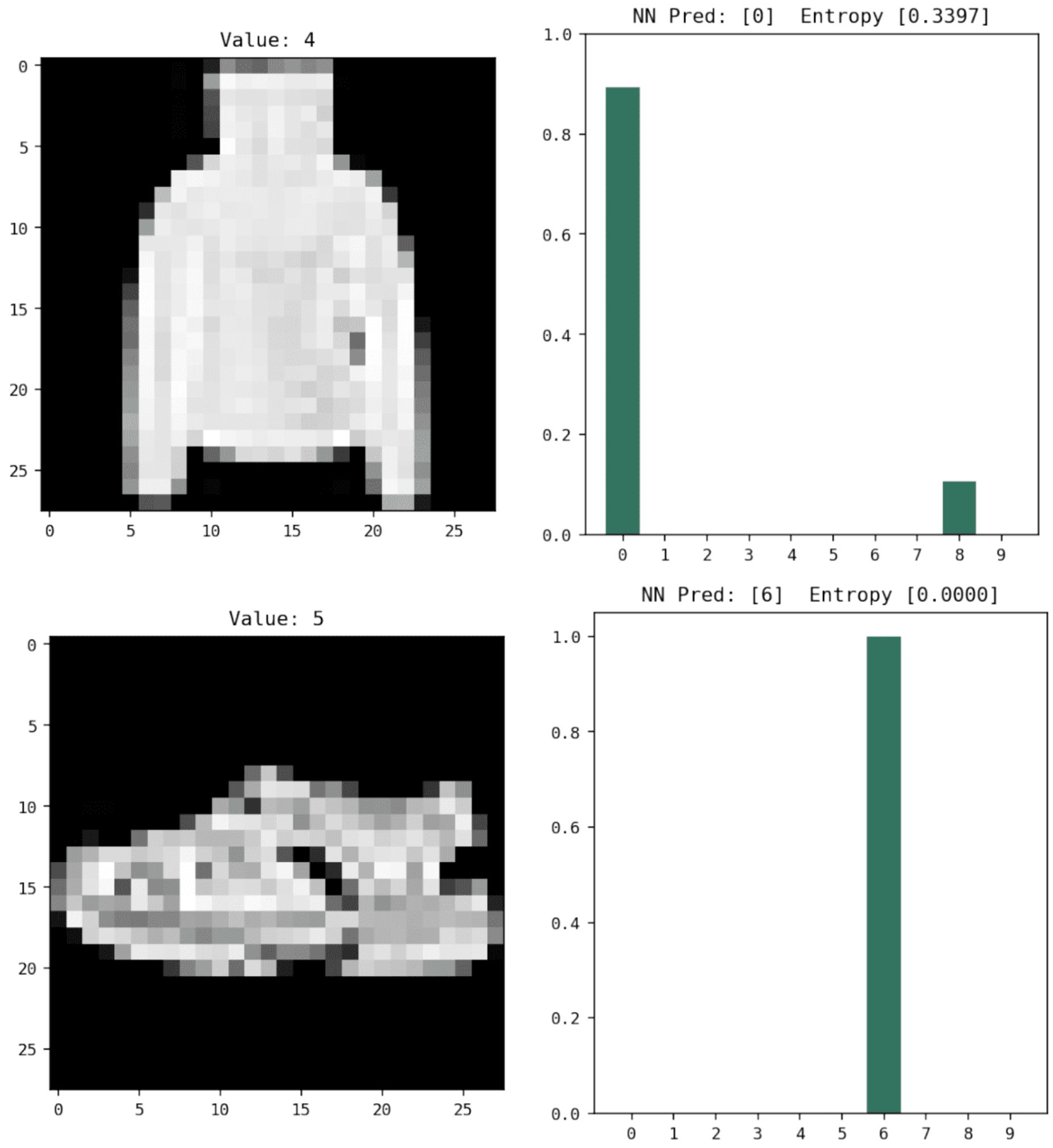
Neural Network are CONFIDENT



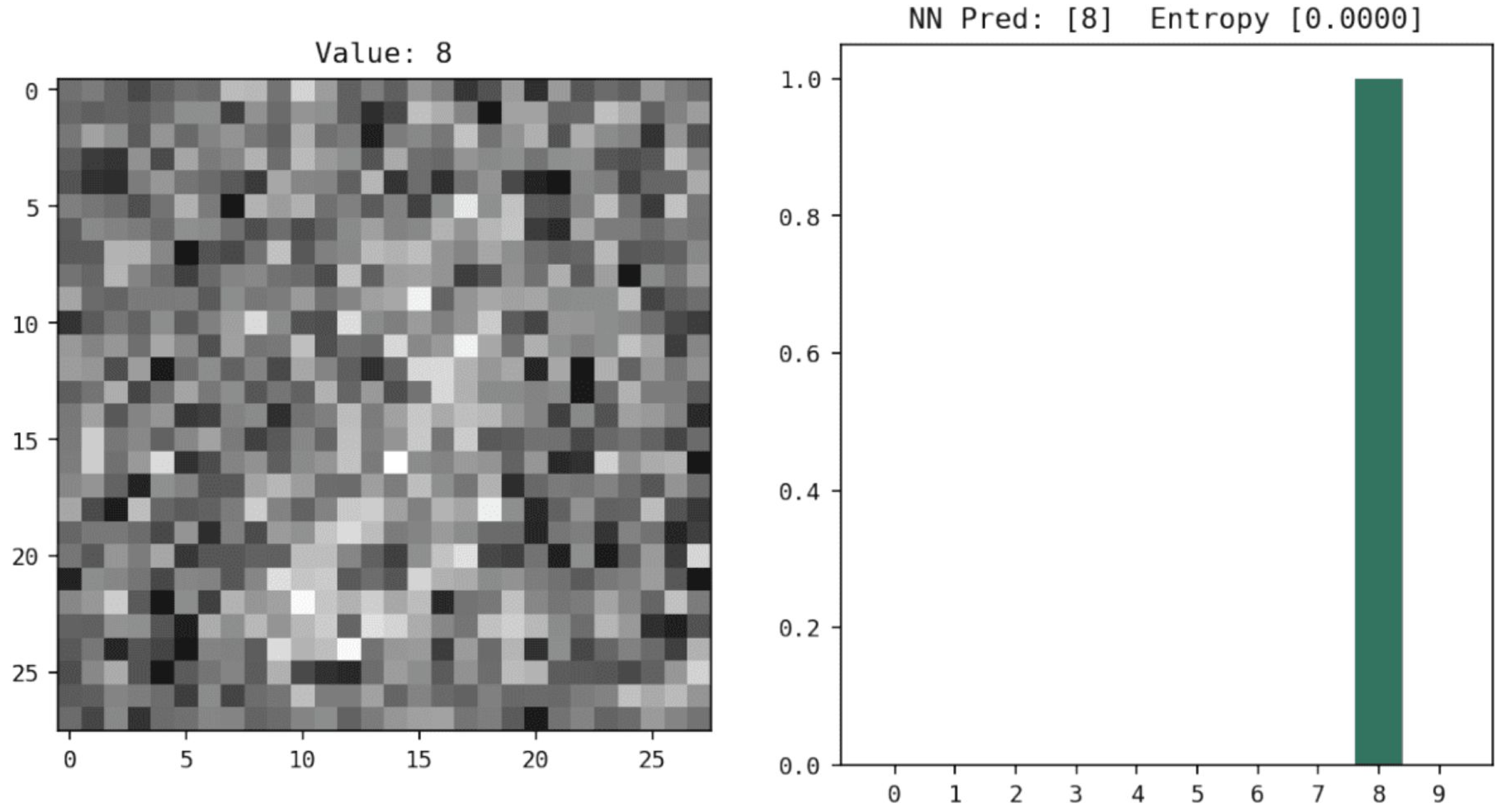
Neural Network are VERY CONFIDENT

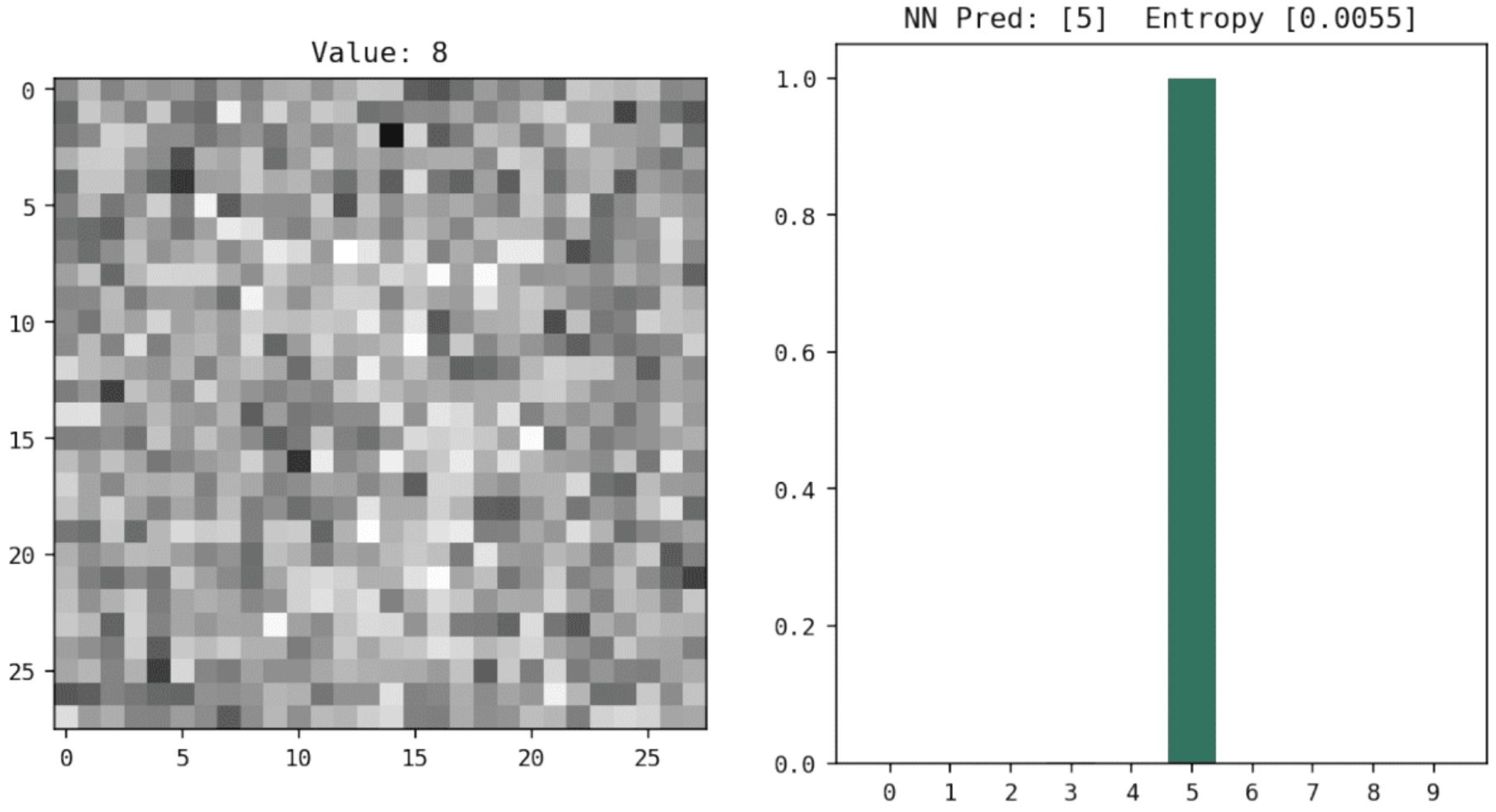


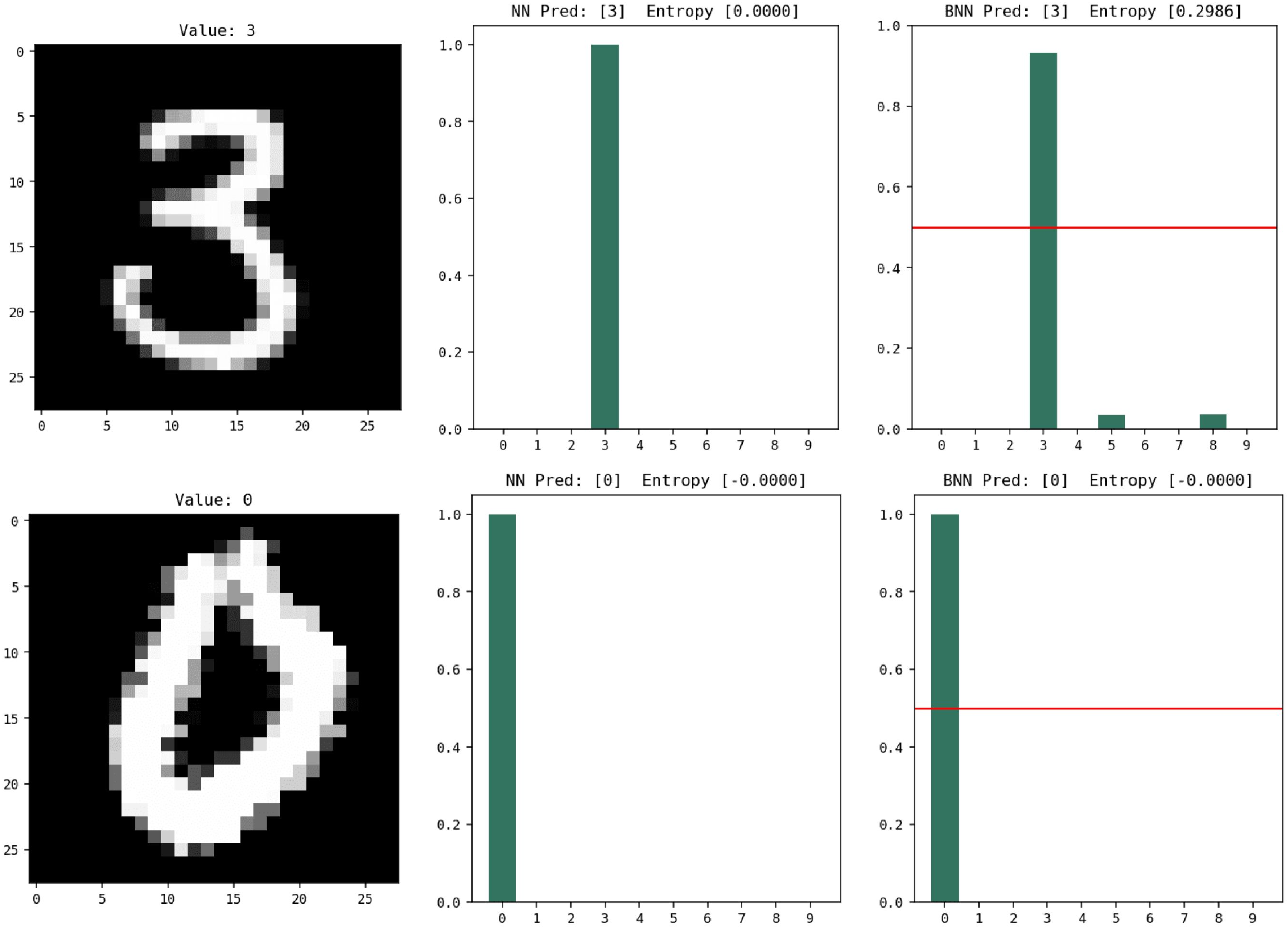
Neural Network are REALLY VERY CONFIDENT



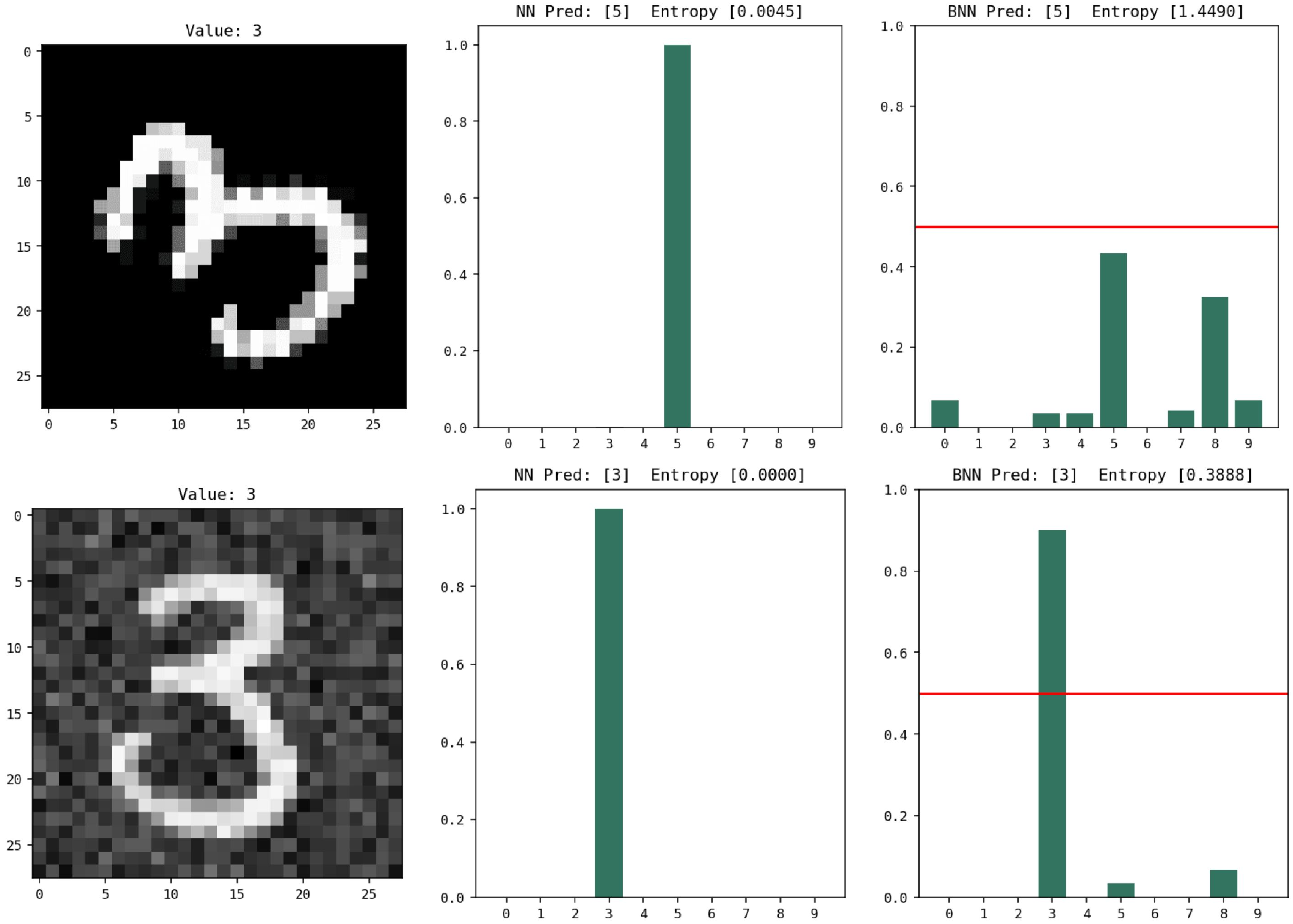
Neural Network
are
OVER CONFIDENT

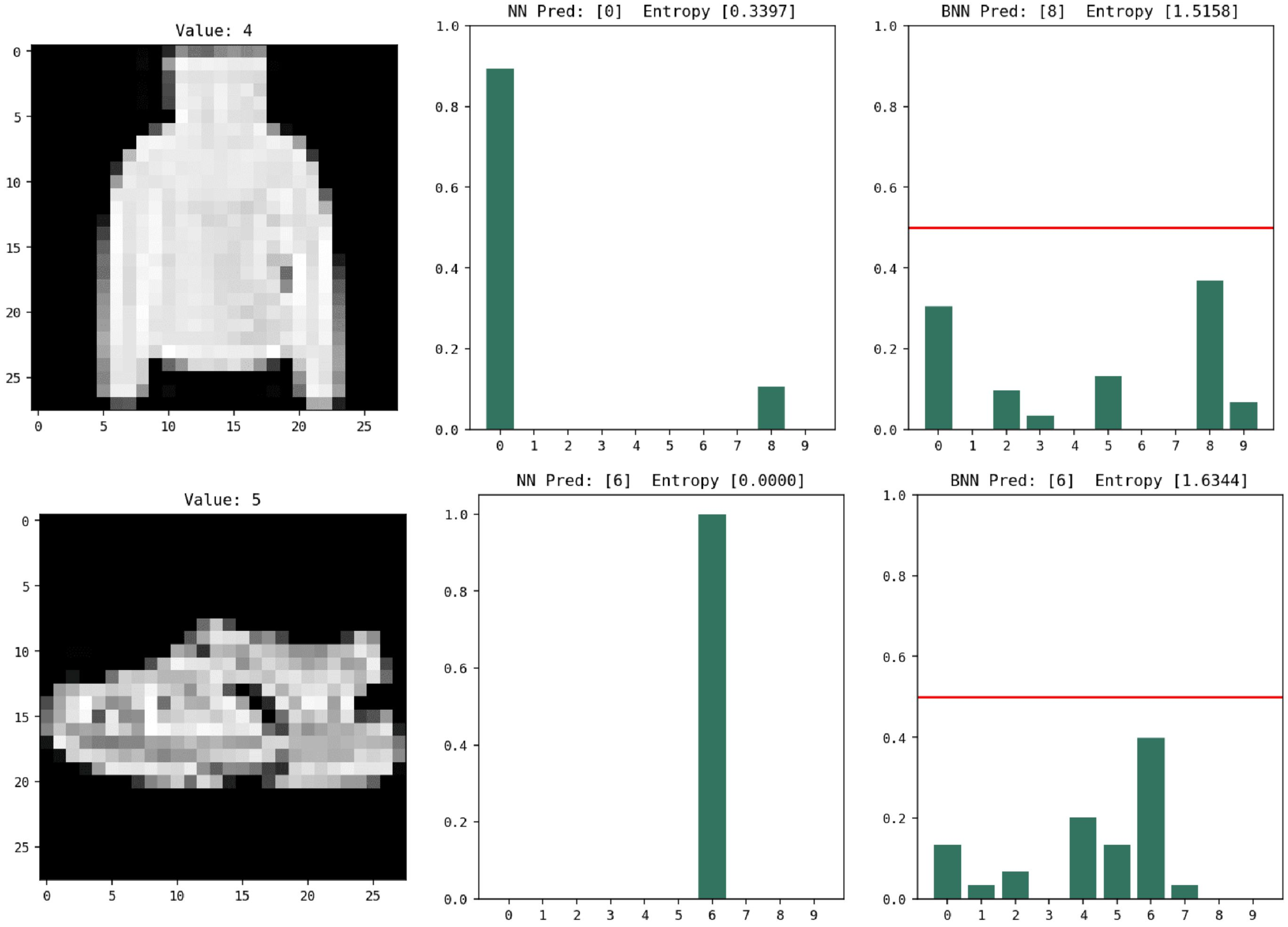




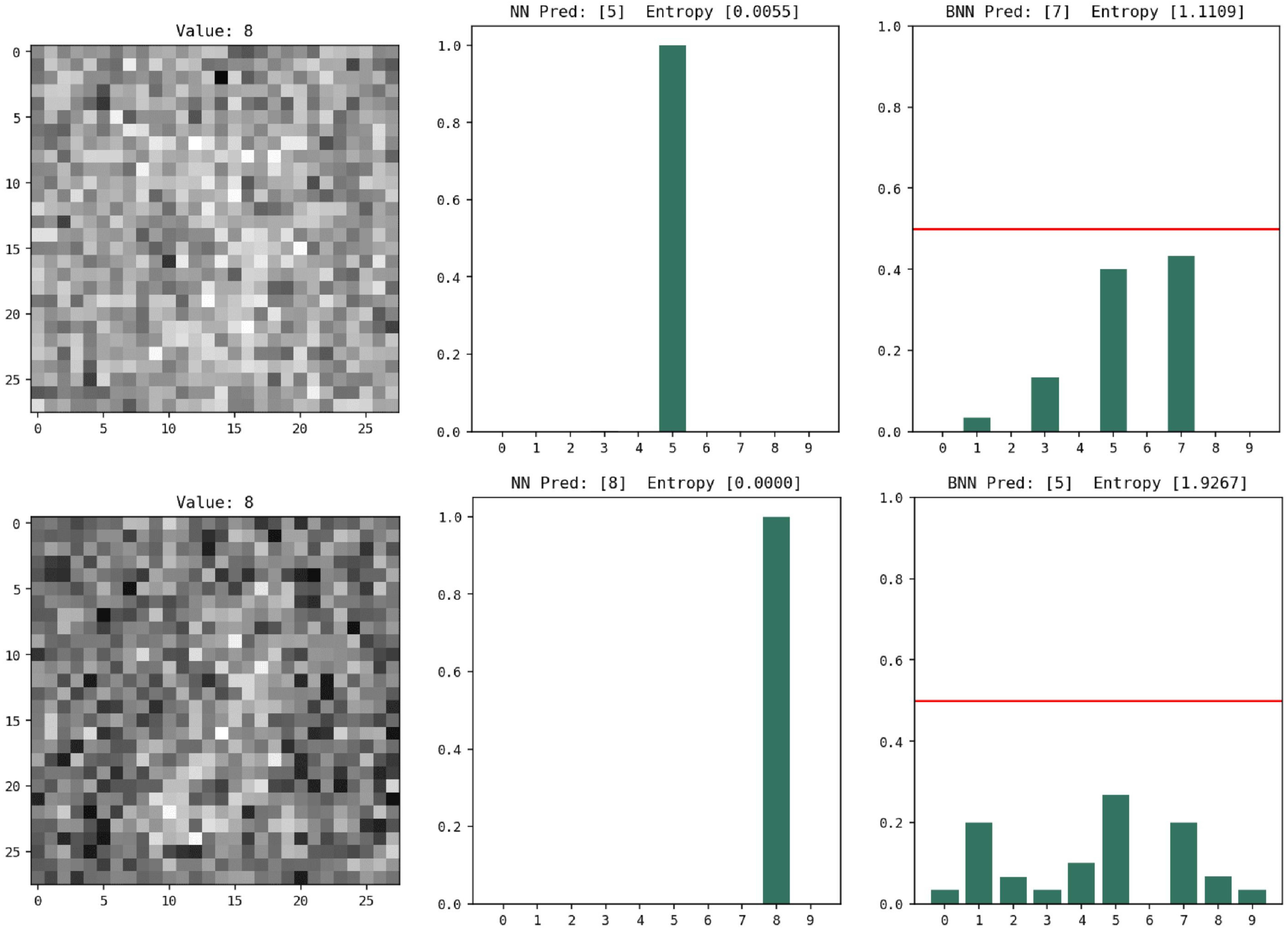


Value: 3
NN Pred: [3] Entropy [0.0000]
BNN Pred: [3] Entropy [0.2986]
Value: 0
NN Pred: [0] Entropy [-0.0000]
BNN Pred: [0] Entropy [-0.0000]


Value: 3
NN Pred: [5] Entropy [0.0045]
BNN Pred: [5] Entropy [1.4490]
Value: 3
NN Pred: [3] Entropy [0.0000]
BNN Pred: [3] Entropy [0.3888]



Value: 4
NN Pred: [0] Entropy [0.3397]
BNN Pred: [8] Entropy [1.5158]
Value: 5
NN Pred: [6] Entropy [0.0000]
BNN Pred: [6] Entropy [1.6344]



Value: 8
NN Pred: [5] Entropy [0.0055]
BNN Pred: [7] Entropy [1.1109]
Value: 8
NN Pred: [8] Entropy [0.0000]
BNN Pred: [5] Entropy [1.9267]

FINALLY,

SKIN LESIONS

MODEL and RESULTS

Lots of NNs tried —— **WE KEPT THIS** ——→

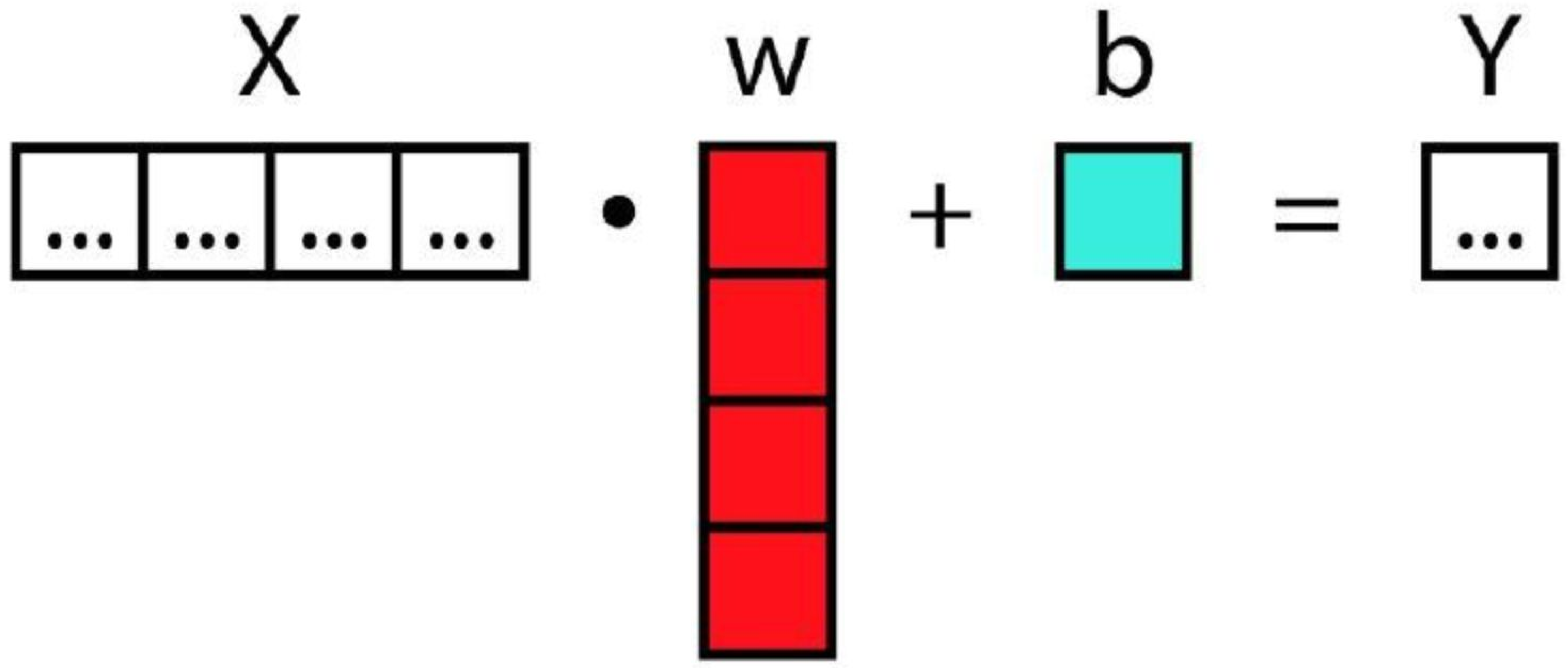
```
nn.Sequential(
 nn.Conv2d(in_channels=3,,out_channels=64,kernel_size=2,),
 nn.ReLU(),
 nn.MaxPool2d(2,2),
 nn.BatchNorm2d(64),
 nn.Conv2d(in_channels=64,,out_channels=512,kernel_size=2,),
 nn.ReLU(),
 nn.MaxPool2d(2,2),
 nn.Dropout(p=0.3),
 nn.Conv2d(in_channels=512, out_channels=1024,,kernel_size=2, ),
 nn.ReLU(),
 nn.MaxPool2d(2,2),
 nn.BatchNorm2d(1024),
 nn.Dropout(p=0.4),
 nn.Conv2d(in_channels=1024, out_channels=1024, kernel_size=1, ),
 nn.ReLU(),
 nn.MaxPool2d(2,2),
 nn.Dropout(p=0.4),
 nn.Flatten(),
 nn.Linear(9216, 256),
 nn.Dropout(p=0.5),
 nn.Linear(256, 7))
```

Grid search applied to find learning rate /momentum

NN	BNN	
CrossEntropyLoss()	TraceMeanField_ELBO()	LOSS
SGD	Adam	OPTIMIZER
70.8%	64.3%	ACCURACY (80-20 split)

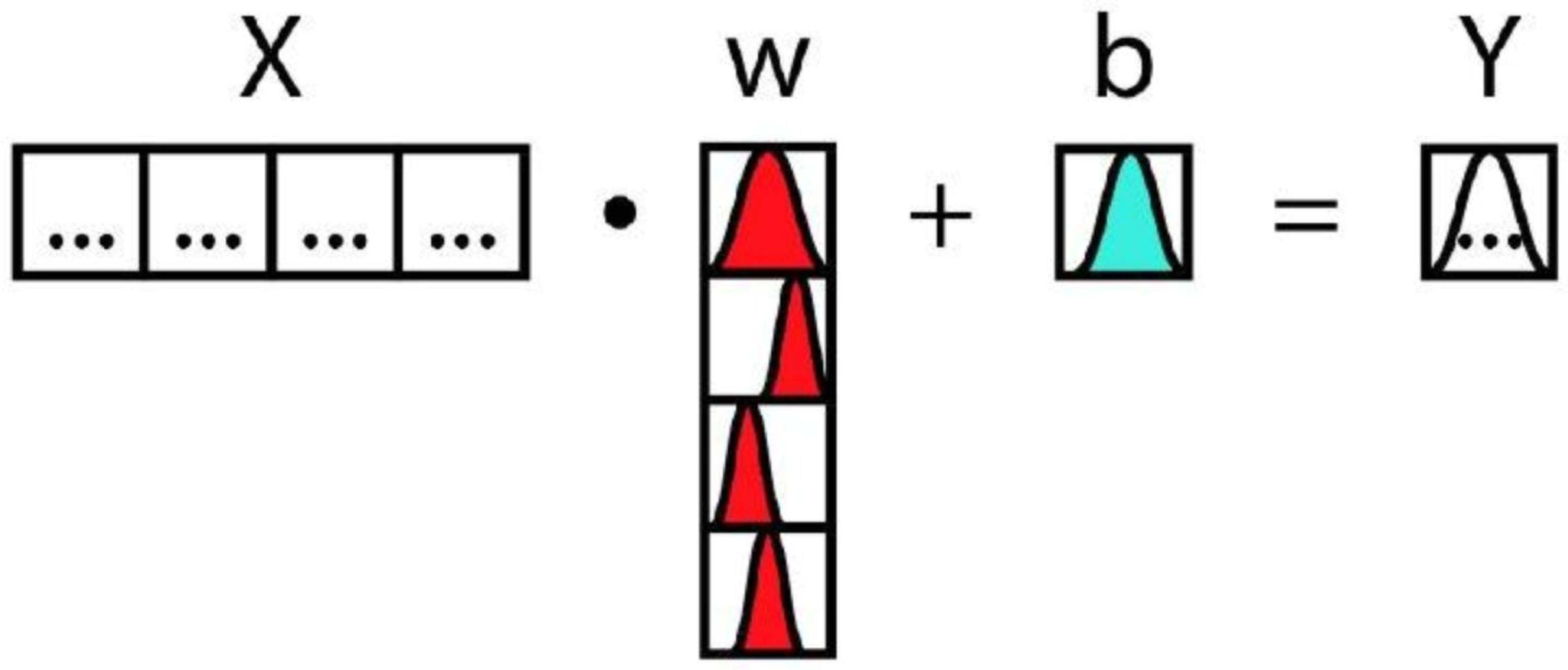
Deterministic Neural Networks

- Deterministic weights and bias



Bayesian Neural Network

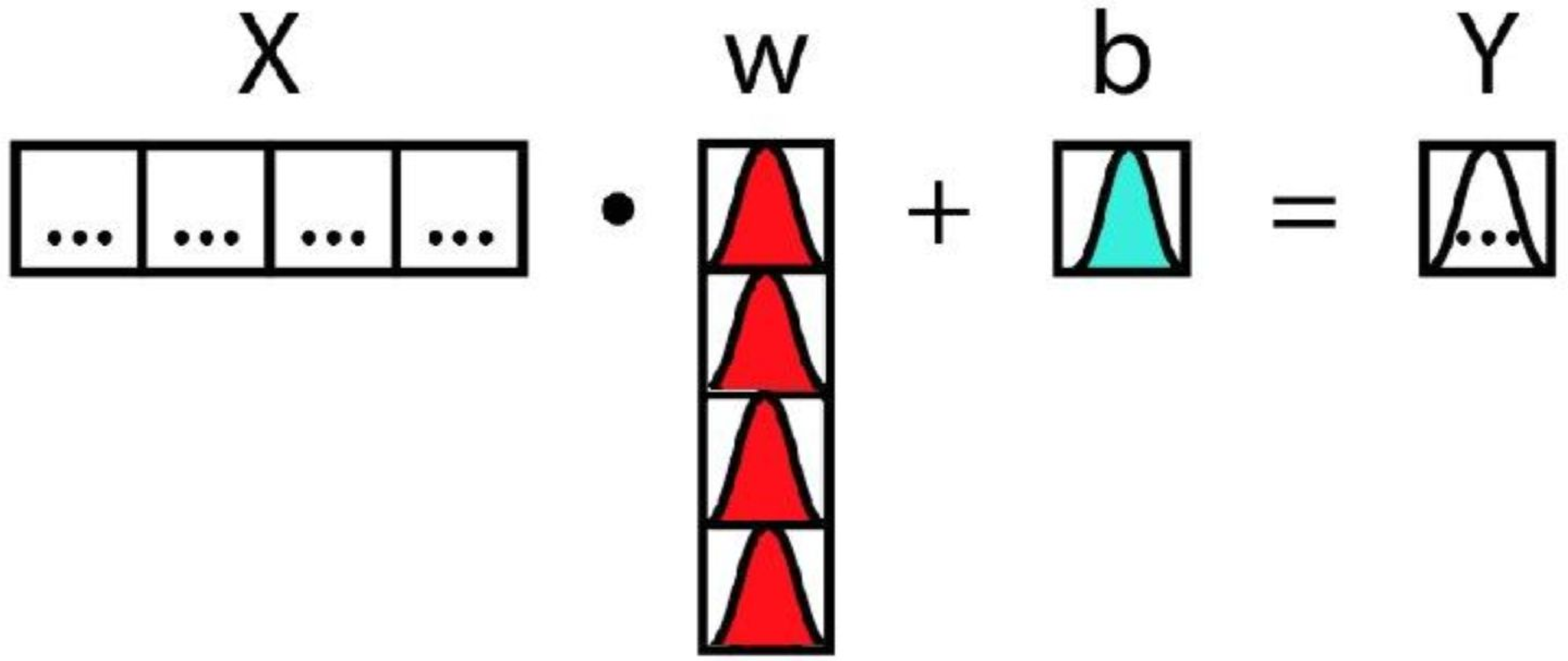
- Stochastic weights and bias



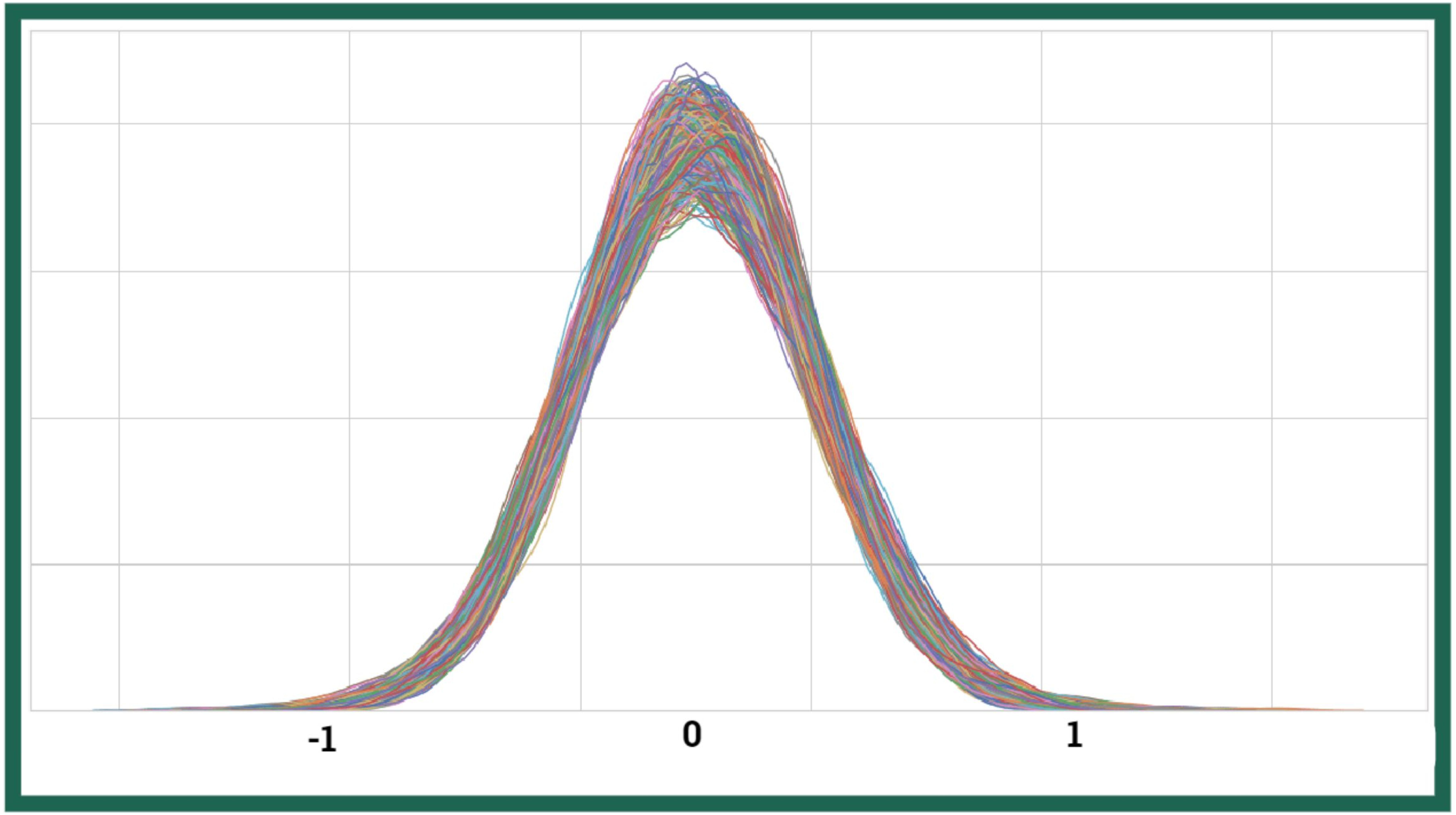
Setting the priors

PYRO CODE:

```
def model(self, x_data, y_data):
     priors = {}
    for key, value in self.net.state_dict().items():
       loc = torch.zeros_like(value)
       scale = torch.ones_like(value)
       prior = Normal(loc=loc, scale=scale)
       priors.update({str(key):prior})
```



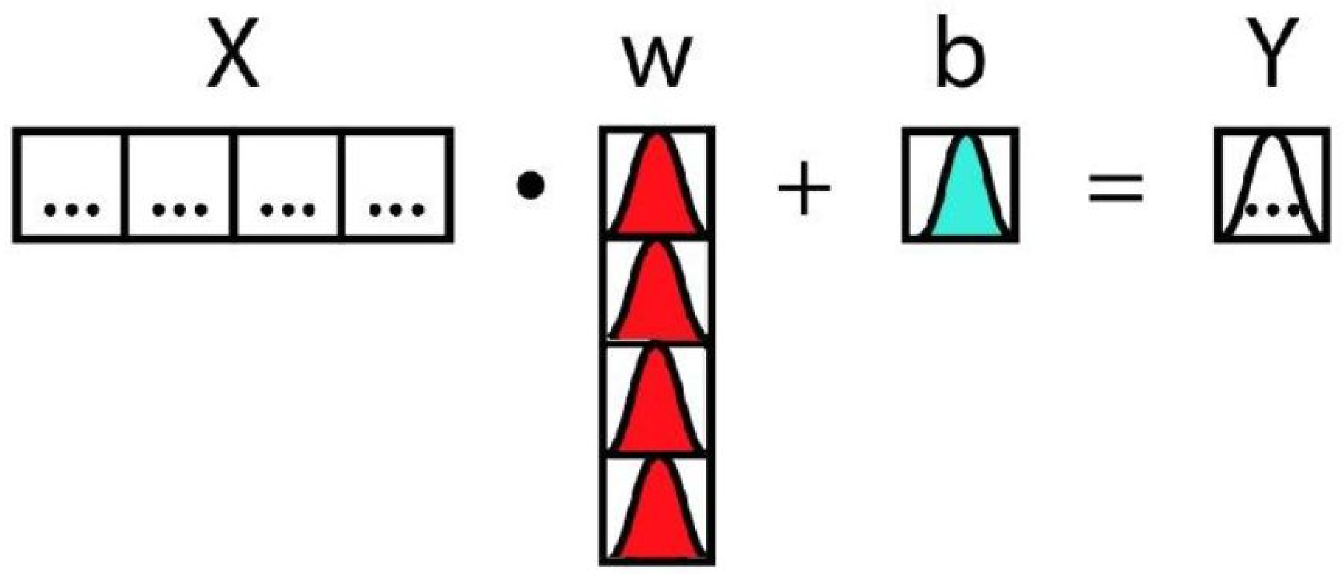
Normal(0,1)



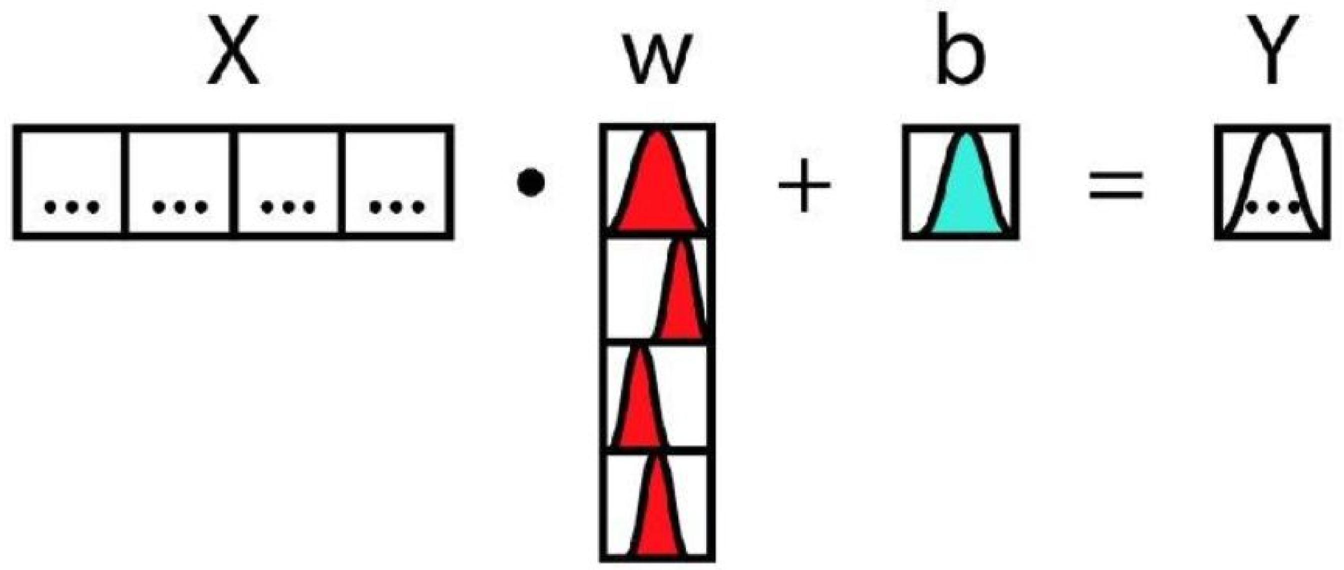
PYRO CODE

```
import seaborn as sns
for w in pyro.get_param_store()['a1_mean']:
      sns.kdeplot(w.detach().numpy())
```

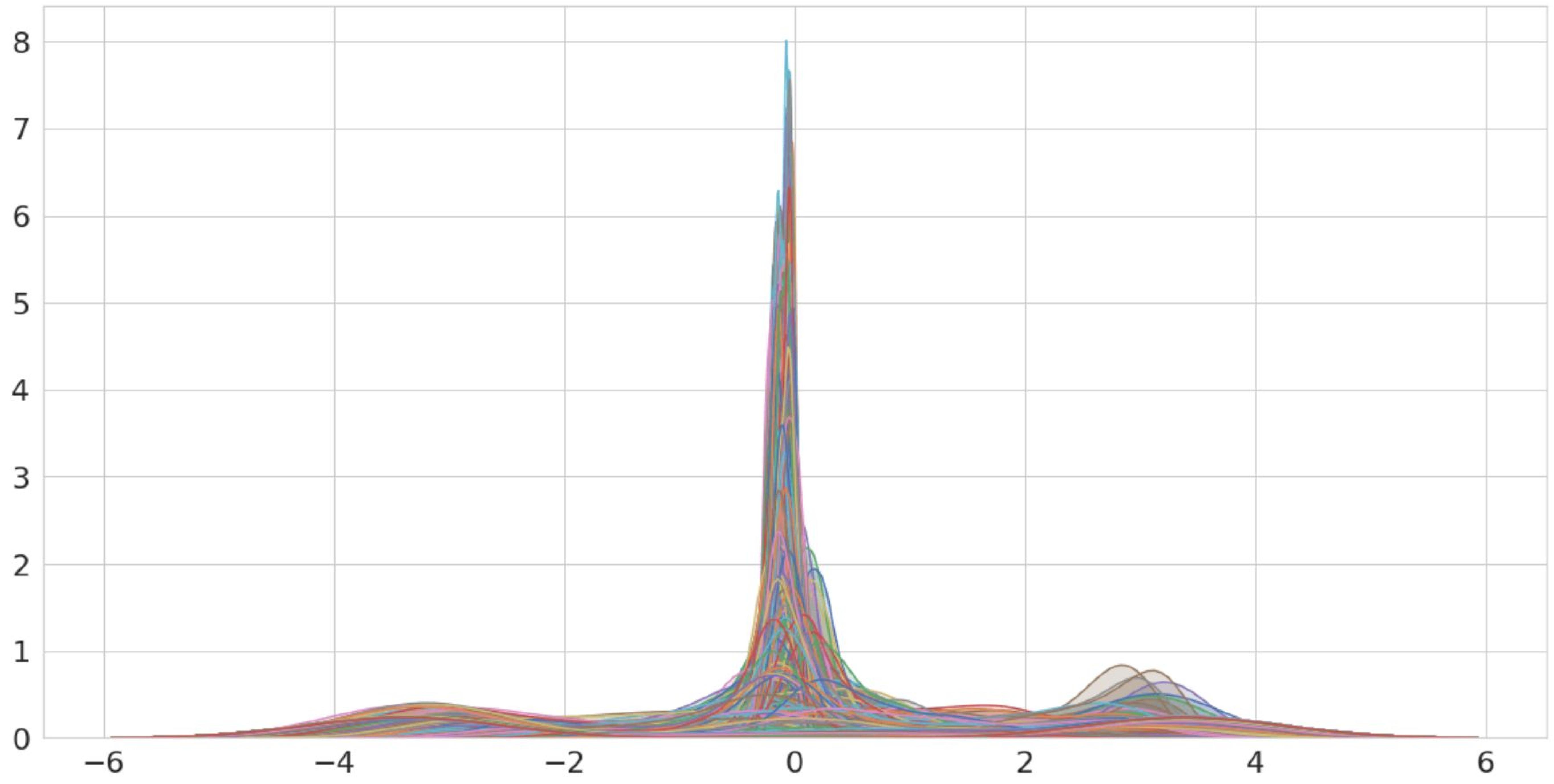
FROM THIS



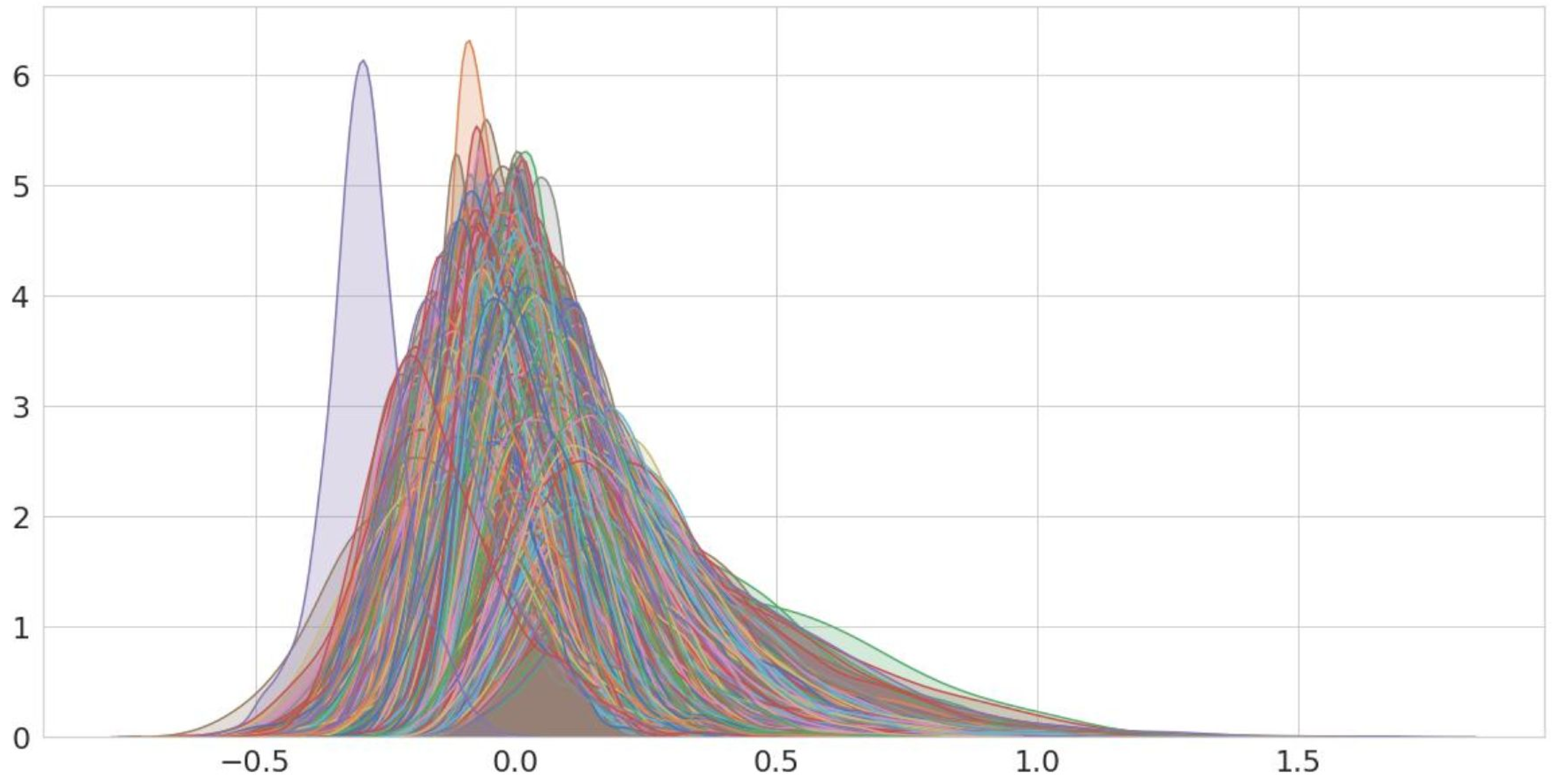
TO THIS

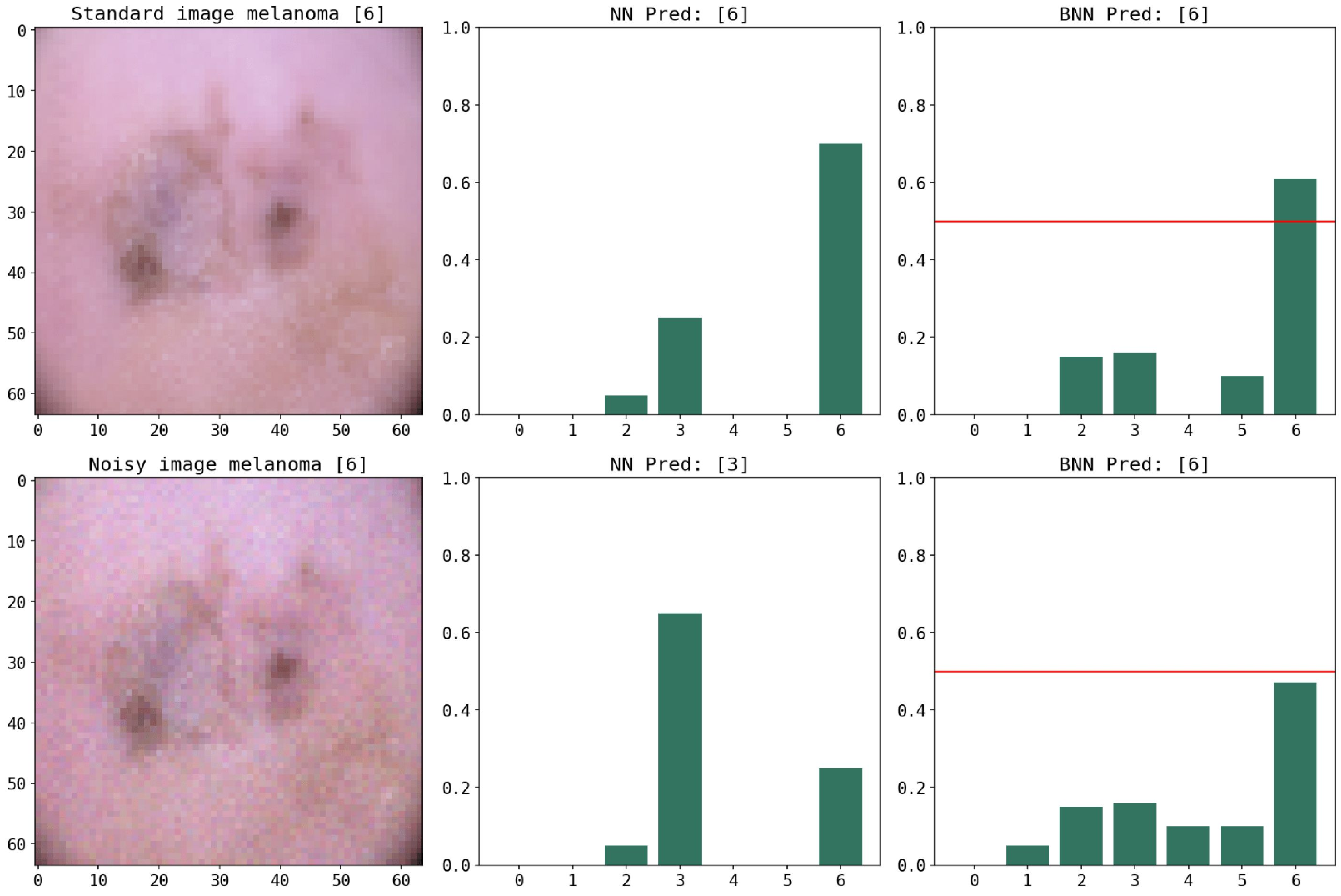


1° LAYER MEANS



1° LAYER SCALES




Standard image melanoma [6]
NN Pred: [6]
BNN Pred: [6]
Noisy image melanoma [6]
NN Pred: [3]
BNN Pred: [6]

CONCLUSIONS

- **Algorithms** for automated diagnostic should take into account **uncertainty** in the estimation
- **Deep Neural Networks** currently represent state-of-the-art for many applications. However, they have some **limitations** and we should not stop trying to improve them.
- **Probabilistic models** can help obtain robust uncertainties which helps to make more informative decisions.
  However, they scale very poorly.
- **Probabilistic Neural Networks** can be a way to combine the best from both philosophies and proved to be more uncertain when observations where out-of-sample/disturbed.

What we would have like to do but we did not have time for :

- Use full resolution images (600 x 450 px)
- GAN-based method for generating synthetic skin lesion images and overcome the unbalanceness of the data
- Experiment deeper NN to increase accuracy
  - Use pre-trained models and make them Bayesian (ResNet-152)

THANK YOU

Paolo Pulcini

Matilde Castelli

DATASET:
SKIN CANCER MNIST: HAM10000 (KAGGLE)

DATA AUGMENTATION:
"Data Augmentation for Skin Lesion Analysis" Perez , Vasconcelos, Avila, Valle

INSPIRATIONAL VIDEOS:
An Attempt At Demystifying Bayesian Deep Learning, Eric J. Ma, PyData NYC 2017
Bayesian Deep Learning with 10 % of the weights, Rob Romijnders, PyData Amsterdam, 2018
Variational Bayes and Beyond: Bayesian Inference for Big Data, Tamara Broderick,2018

CONVOLUTIONAL NEURAL NETWORK:

"Automated classification of cells into multiple classes in epithelial tissue of oral squamous cell carcinoma using transfer learning and convolutional neural network", Das, Hussian, Mahantam, 2020

BAYESIAN NEURAL NETWORK:

"Robust Bayesian Neural Networks", Ginevra Carbone (GitHub)
"Bayesian Sparsification of Neural Networks", Nadezhda Chirkova (GitHub)
"Bayesian CNN", kumar-shridhar (GitHub)
"Bayesian Convolutional Neural Networks" Yarin Gal, Zoubin Ghahramani

GAN:
"Skin lesion synthesis with GAN" Perez , Vasconcelos, Avila, Valle