ECE695 – ILGM

Bayesian Inference using Neural Networks

By: Sheikh Shams Azam

04/26/2021

Outline

- Motivation
- Neural Network Training
- HMM of Weights: Kalman Filter
- Non-Gaussian Weight Initialization: Particle Filter
- Summary
- References

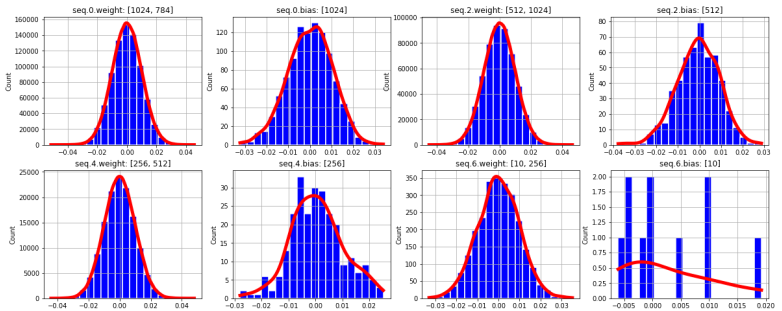
Motivation

- Neural Networks (NN) fail with confidence, i.e., false predictions with high confidence.
- Solutions to this generally involves ensembling of multiple models trained independently.
- In the optimization process of NNs, we discard the steps taken and only store the end weights.
- Proposal: The path taken by the optimization can help us learn distribution of weights.
- The parameter distribution can help us draw several models.
- So basically, we can try to do a Bayesian prediction using NNs.

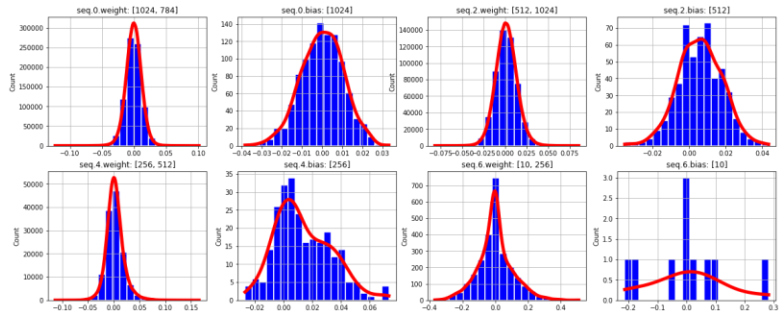
$$\underbrace{q(X_{n+1}|\theta^*)}_{\text{Frequentist}} \quad \textbf{vs} \quad \underbrace{\int_\theta q(X_{n+1}|\theta)q(\theta|x)d\theta}_{\text{Bayesian}}$$

Neural Network Training

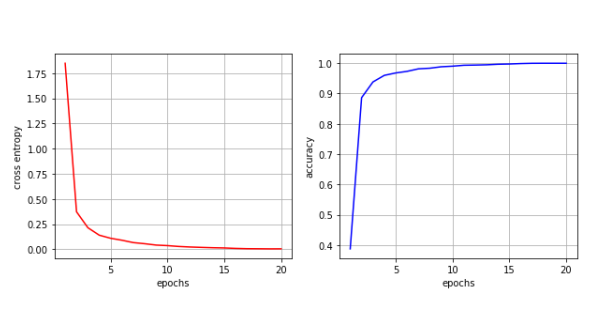
- Often NNs initialized using standard Gaussian distributions, $\theta_0 \sim \mathcal{N}(\mu_{\theta_0}, \sigma^2_{\theta_0})$.
- Training NN with gradient descent is a linear update: $\theta_{k+1} = \theta_k - \eta \cdot \nabla f_{\theta_k}(X, Y)$.
- A Gaussian distributed RV undergoing a linear (affine) transformation gives another Gaussian.



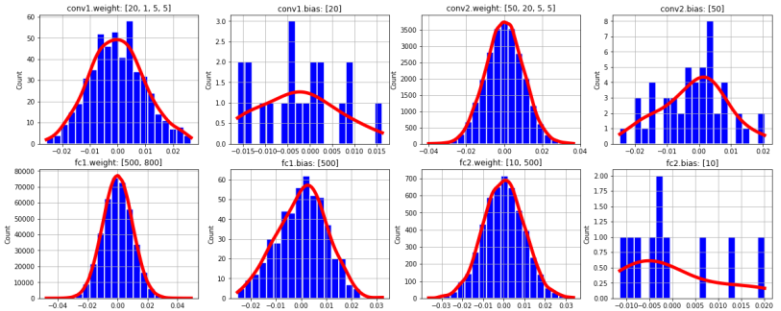
FCN: initialization



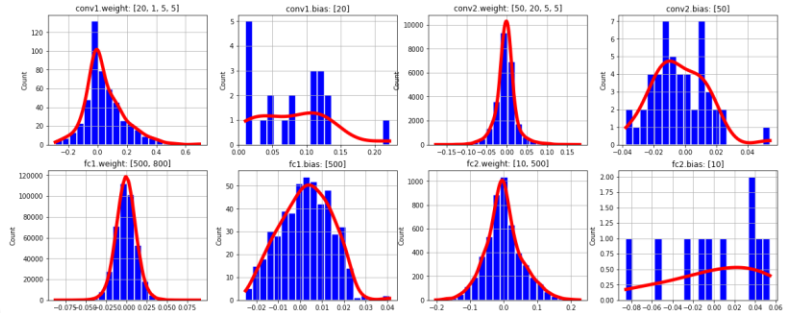
FCN: epoch 20



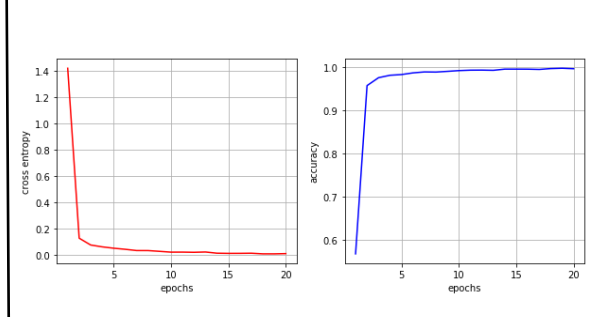
FCN: training



CNN: initialization



CNN: epoch 20



CNN: training

HMM of Weights: Kalman Filter

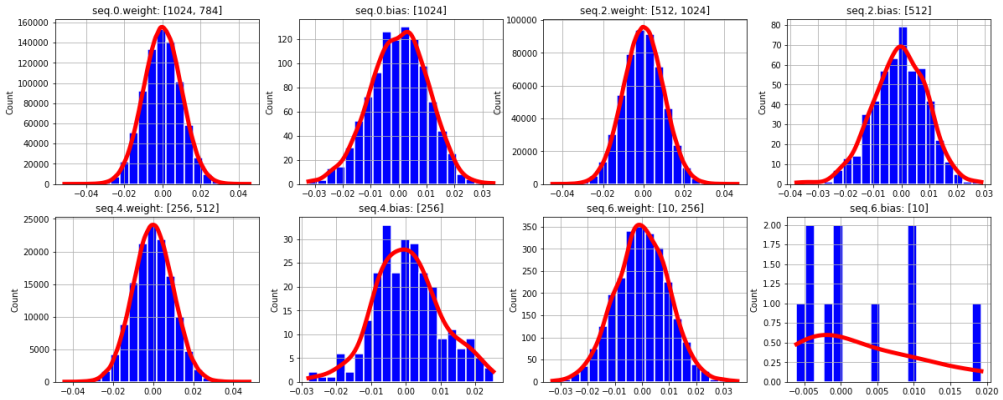
- Means of the weights follow a Markov process:

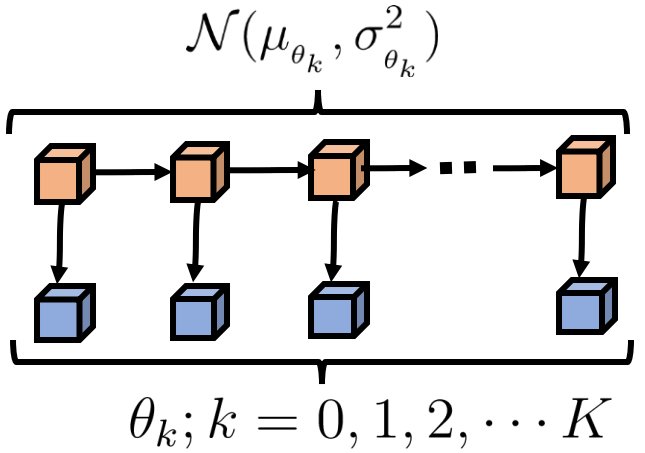
$$\mu_{k+1} = \mu_k - \mathbb{E}[\eta \cdot \nabla f_{\theta_k}(X, Y)]$$

- Similarly, variance of weights can be modelled:

$$\sigma^2_{k+1} = \sigma^2_k + \eta^2 \mathbb{E}[(\nabla f_{\theta_k})^2] - \eta^2 \mathbb{E}^2[\nabla f_{\theta_k}]$$

- Tracking the means becomes a direct application of Kalman filtering in this case.
- Since the emissions are Gaussian distributed we get a recursive solution for tracking the means and variance.
- Simplifying assumptions:
  - Weights in different layers are independent of each other
  - Weights within the same layer can be considered pairwise related in which case it is possible to track the covariance matrix.

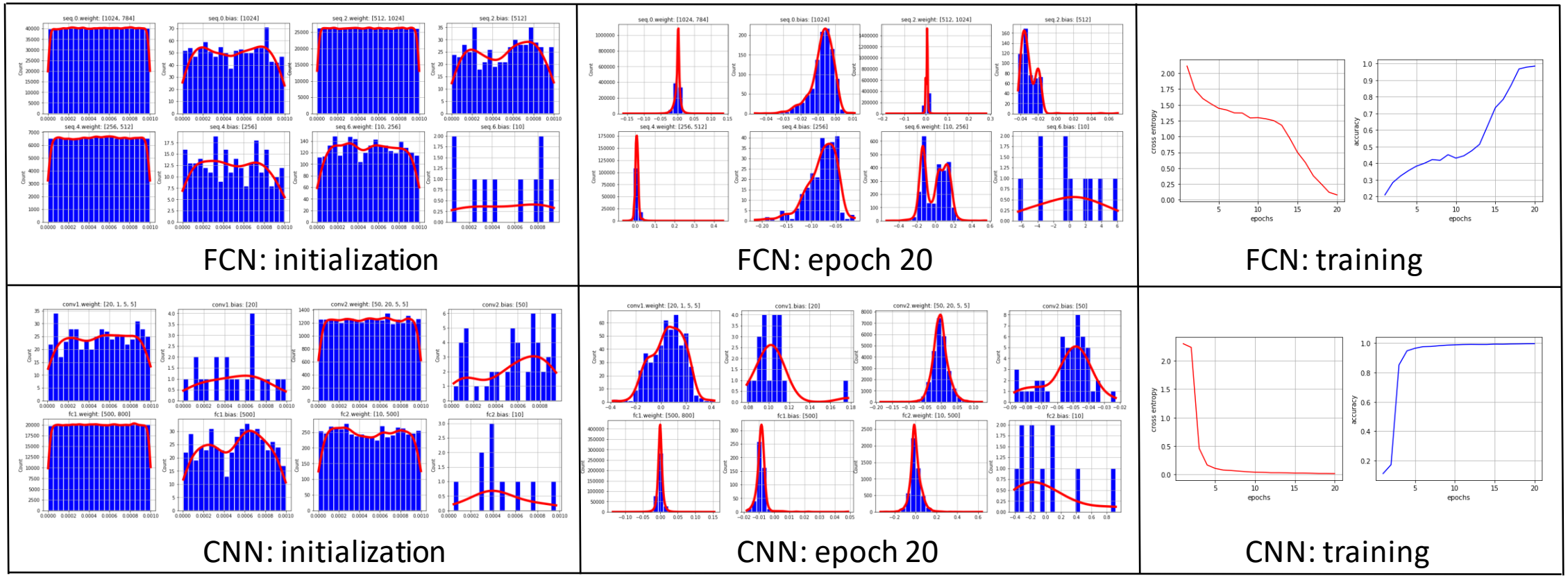




Reference: Franchi, Gianni, et al. "TRADI: Tracking deep neural network weight distributions." *ECCV 2020*.

Non-Gaussian Weight Initialization: Particle Filtering

- NNs can also be initialized using non-Gaussian distributions.
- Simple case of uniform distribution. Final distributions are non-Gaussian like.
- Recursive Kalman-Filter is not the best fit. So, we look at Particle filter instead.



Summary

- We can do Bayesian inference by tracking the weight distribution instead of just the final weights.
- We can also quantify the uncertainty in prediction by drawing and ensembling multiple models.
- KF is a natural way to go if the initializations are Gaussian and transformations are linear.
- But, PF would generalize better to cases where Gaussian or linearity assumptions are violated.
- Memory requirement of saving ensembles is higher than weight distribution tracking.
- There are other methods like deep-ensemble methods that have higher training cost as well.
- There are pros and cons of KF vs PF as well:
  - KF is faster because Gaussian assumption leads to recursion.
  - PF on the other hand is extendable to different types of networks.
  - PF can work with any initialization: sparse, orthogonal etc.
  - PF works with non-linear transformations in optimization: second order or meta-learning.

References:

- Implementation for this work (to be updated): https://github.com/shams-sam/EE695-ILGM.
- Kalman and Bayesian Filters in Python: https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python is a good resource for introduction to other filtering approaches.
- Franchi, Gianni, et al. "TRADI: Tracking deep neural network weight distributions." *European Conference on Computer Vision (ECCV) 2020*. (code for this paper is not released; vectorized implementation is available in my project: https://github.com/shams-sam/EE695-ILGM).