ECE695 – ILGM

1. Kalman Filter with Neural Networks
2. Variational Autoencoders with Normalizing Flows

By: Sheikh Shams Azam

04/26/2021

Kalman Filter with Nueral Network

Motivation

- Current research in studying the optimization subspace in gradient descent.
- Came across this paper [1], One line summary:

  "gradient descent tell us more than what we care about".

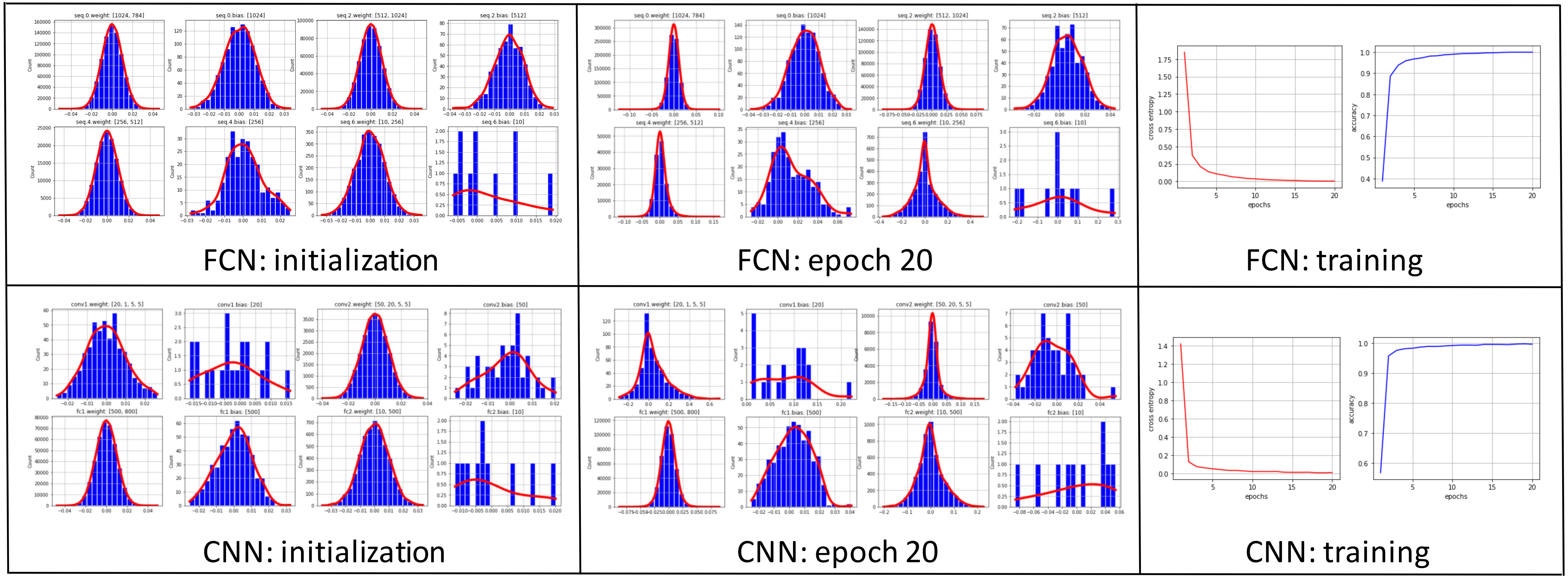
- Idea:
  - Neural Networks (NN) fail with confidence, i.e., false predictions with high confidence.
  - Solutions to this generally involves ensembling of multiple models trained independently.
  - In the optimization process of NNs, we discard the steps taken and only store the end weights.
  - Proposal: 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.

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

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

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.



HMM of Weights: Kalman Filter

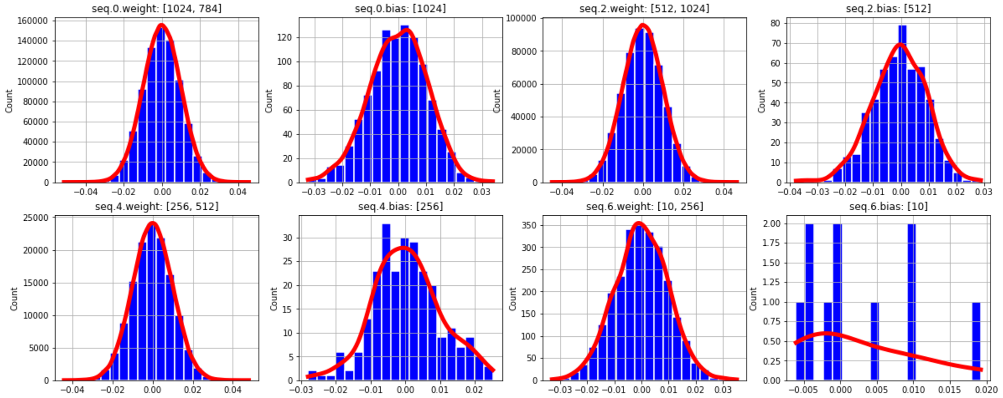
- Means of the weights follow a Markov process:

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

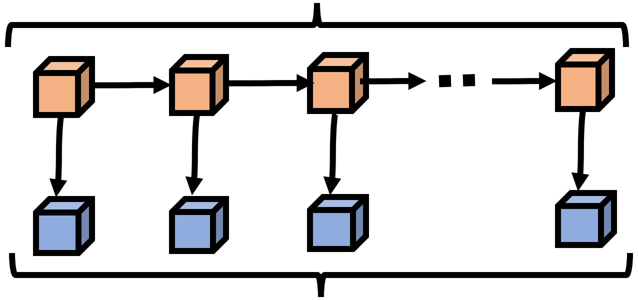
- Similarly, variance of weights can be modelled:

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

- 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.



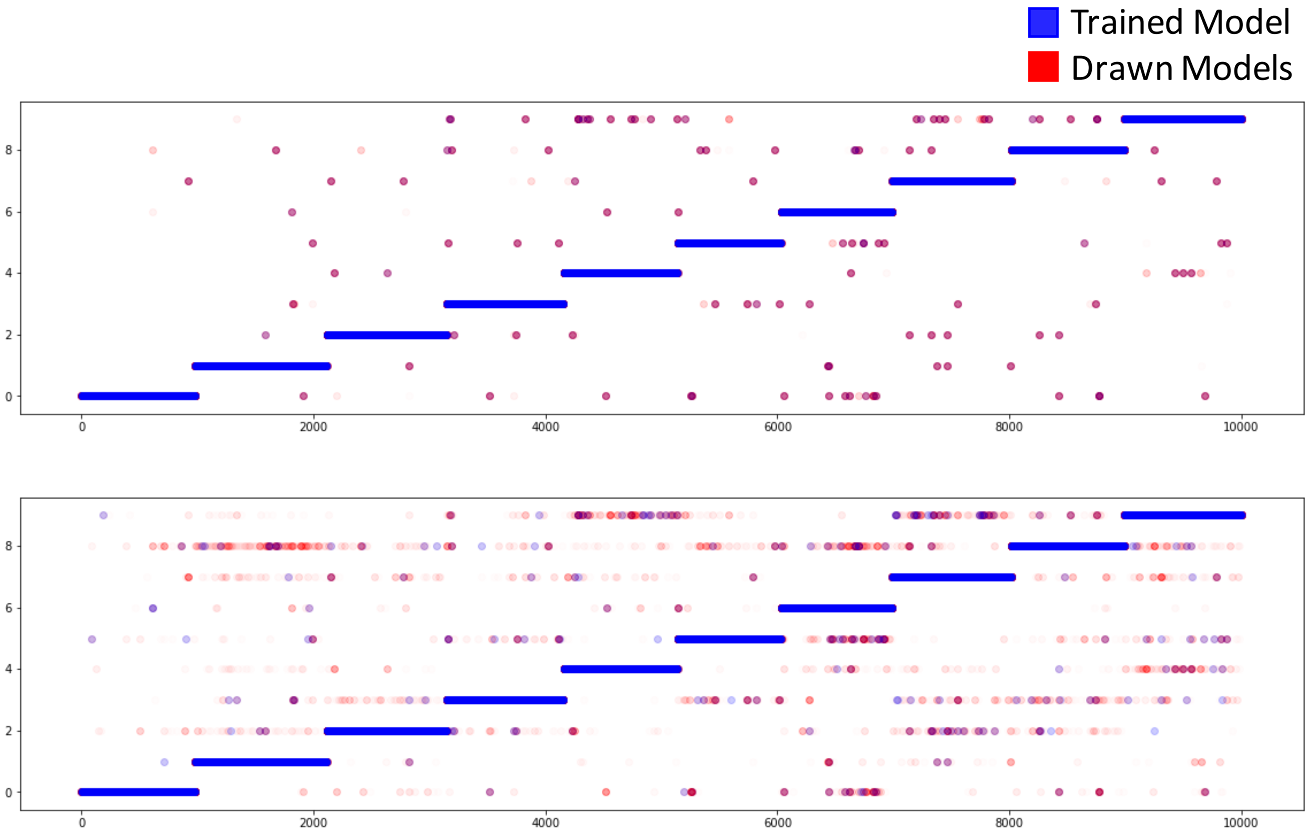
$$\mathcal{N}(\mu_{\theta_k}, \sigma^2_{\theta_k})$$



$$\theta_k; k = 0, 1, 2, \cdots K$$

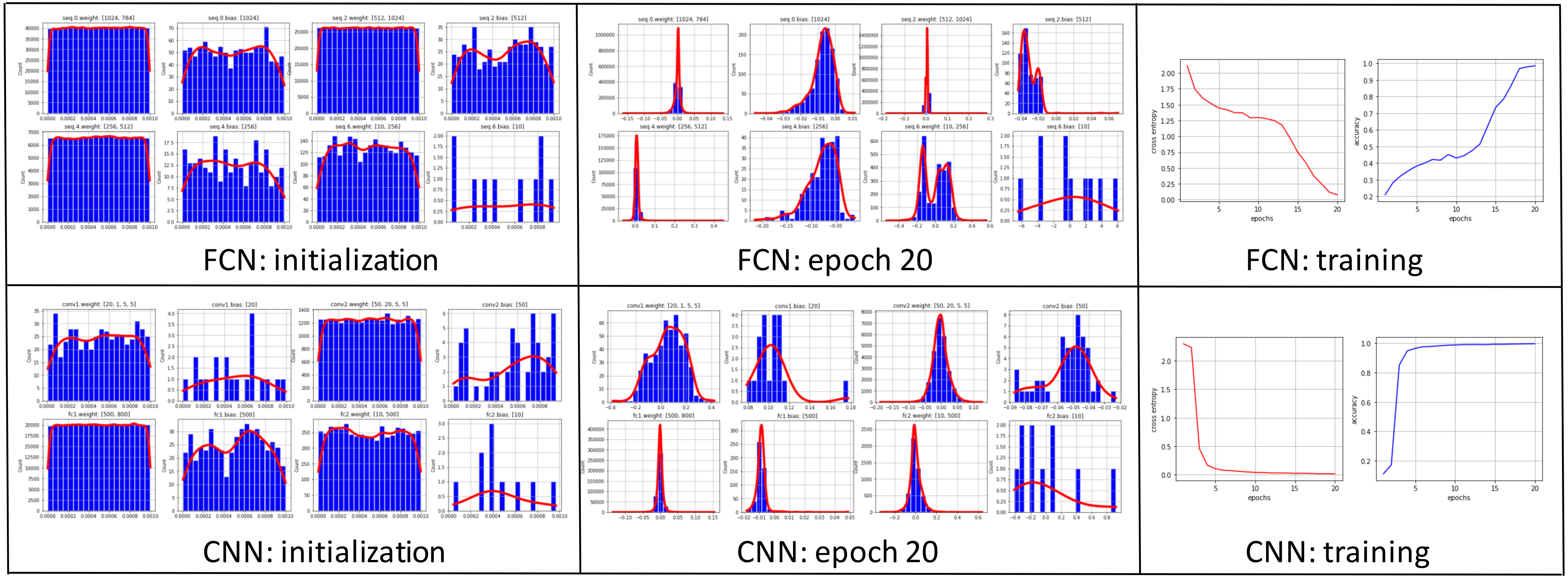
Some Results: Kalman Filter

- Trained model gives the frequentist prediction
- Models drawn from the probabililty distribution give better picture about the uncertainty in prediction.
- Classification on noisy data (bottom graph) gives a good picture of uncertainty in prediction.
- Can be used to detect out of distribution samples by checking the variance in prediction.
- There are other methods like deep-ensemble, MC Dropout methods in the same domain.
- Memory requirement of saving ensembles is higher than weight distribution tracking.



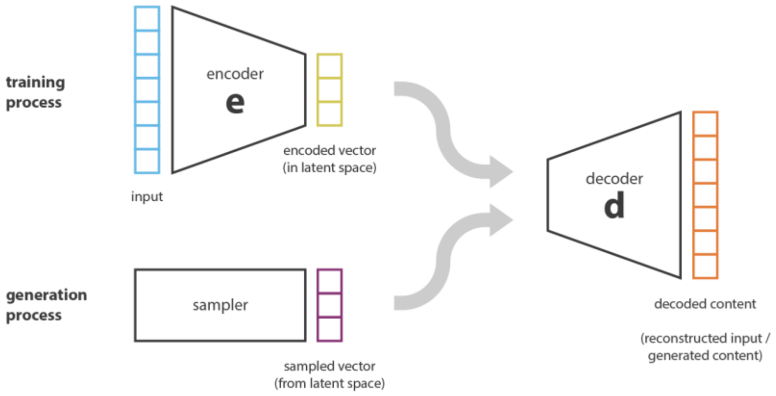
Non-Gaussian Weight Initialization

- NNs can also be initialized using non-Gaussian distributions.
- Simple case of uniform distribution. Final distributions are non-Gaussian like.
- Kalman-Filter could be replaced with Particle filter instead but computation overhead.



Variational Autoencoder with Normalizing Flow

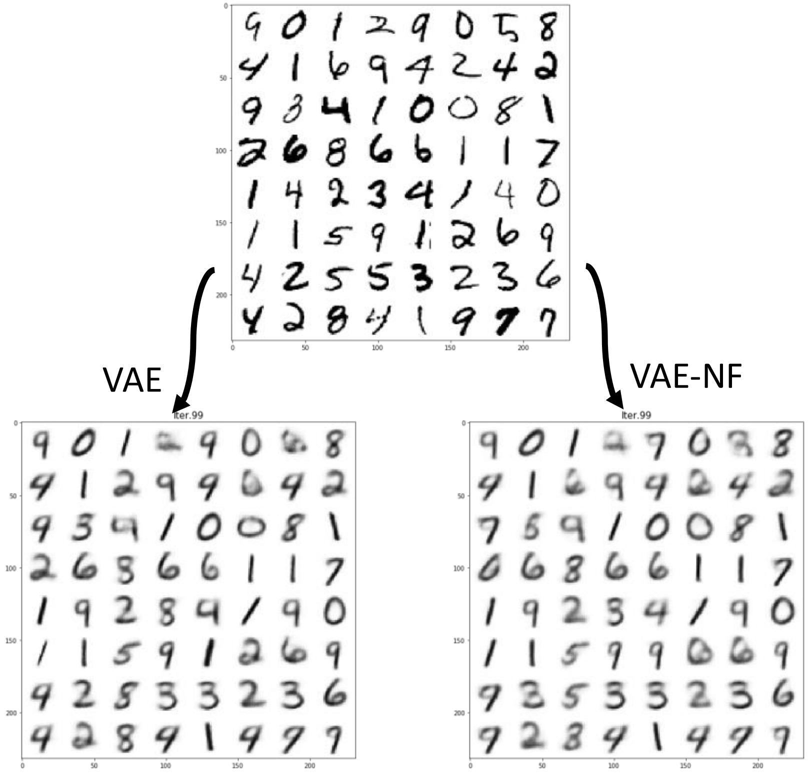
Normalizing Flows with VAE



$$\underbrace{\mathbb{E}_{r(X|Y)}[-log\, q(X|Y)]}_{\text{Reconstruction loss}} - D_{KL}[r(X|Y;\phi)||q(X;\theta)]$$

VAE: The posterior is approximated with a shifted and scaled Gaussian (decouples sampling and X).
Normalizing flows can relax this simplification!!!

$$\mathbb{E}_r[log\, r(X|Y) - log\, q(X,Y)] = \mathbb{E}_r[\underbrace{log\, q_K(X_K)}_{\text{Flow Output}} - log\, q(X,Y)]$$



References:

- Implementation for this work (final update soon): 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).
- Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." *arXiv preprint arXiv:1312.6114* (2013).
- Kobyzev, Ivan, Simon Prince, and Marcus Brubaker. "Normalizing flows: An introduction and review of current methods." *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

Questions?