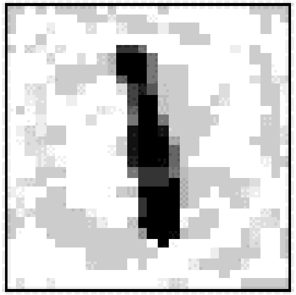
Extended Variational Inference for Propagating Uncertainty in Convolutional Neural Networks

DIMAH DERA, GHULAM RASOOL AND NIDHAL BOUAYNAYA

Department of Electrical and Computer Engineering
Henry M. Rowan College of Engineering

How Robust Are Machine learning systems?

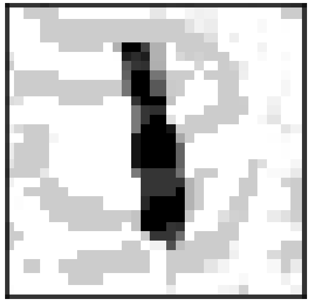
- Mistakes done by lower-level machine learning components can propagate up the decision making process and lead to devastating results.
- Systems where decision-making and control is handed over to autonomous systems include
- autonomous control of drones and self-driving cars
- healthcare diagnosis
- high-frequency trading



True: 1, Pred: 3

OUTPUT OF SOFTWMAX IS NOT A PROBABILITY

EXAMPLE: 0.2 ADVERSARIAL NOISE (MISCLASSIFIED INPUT)



True: 1, Pred: 3

Ground Truth (yellow)
Network Prediction (green)

Prediction of Deterministic CNN

2E-13	**0**
0.0238	**1**
8E-08	**2**
0.9762	**3**
7E-10	**4**
5E-08	**5**
5E-10	**6**
1E-09	**7**
8E-06	**8**
2E-13	**9**

Hand-engineered features in Computer Vision

original graylevel image

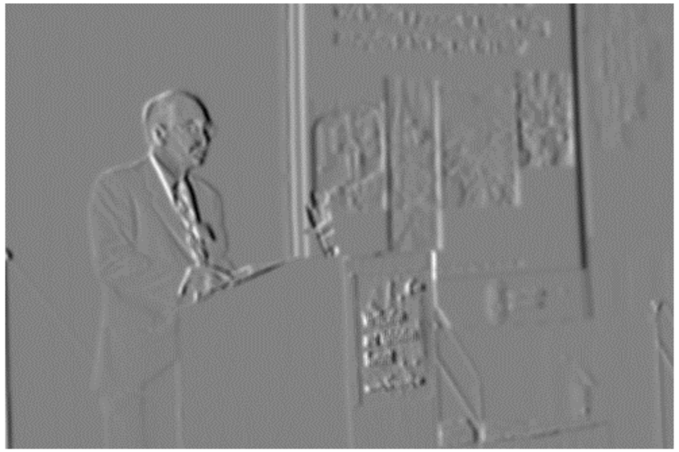
*
Convolution

1	0	-1
1	0	-1
1	0	-1

Vertical edge filter

=

Vertical edge responses

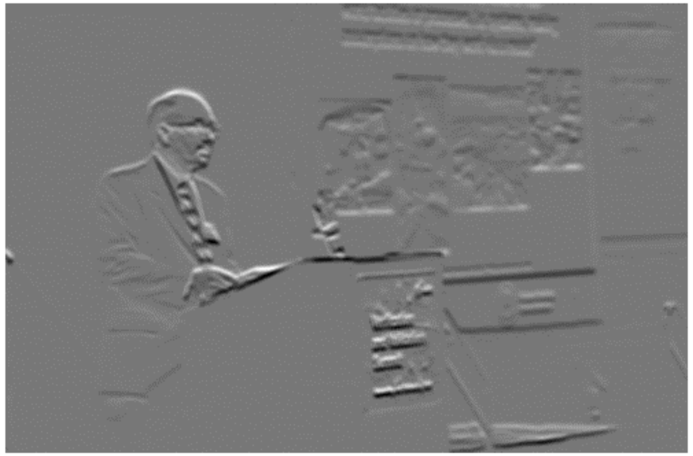


1	1	1
0	0	0
-1	-1	-1

Horizontal edge filter

=

Horizontal edge responses



- Also, try Sobel filter, Scharr filter, Canny edge detector, Gaussian filters, Gabor filter, etc

IDEA OF CONVOLUTIONAL NEURAL NETWORKS – LEARN THE FEATURE KERNELS

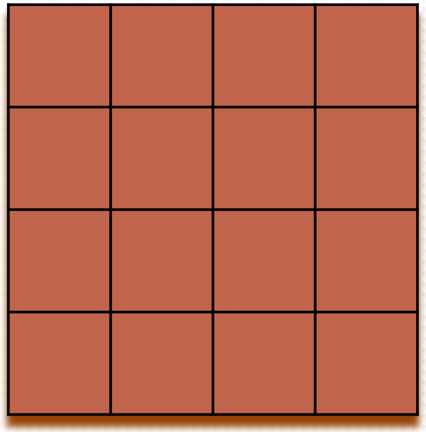
original graylevel image

*

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

(Unknown) Kernel

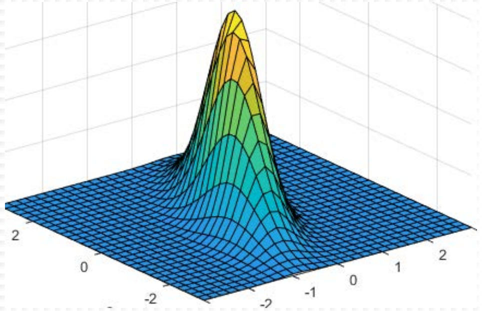
=



A person giving a presentation

UNCERTAINTY ESTIMATION – BAYESIAN DEEP LEARNING

*



w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

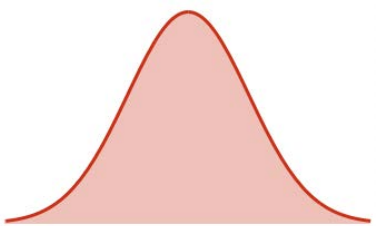
(Unknown) random Filter

=

Mike Paglione giving a presentation to an audience

+

How confident the model is in its inference



Bayesian Inference with Tensor Normal Distributions

- Bayesian inference for neural networks captures uncertainty in the weight parameters

$$\boldsymbol{\Omega} = \left\{ \left\{ \{\mathcal{W}^{k_c}\}_{k_c=1}^{K_c} \right\}_{c=1}^{C}, \{\mathbf{W}^{(l)}\}_{l=1}^{L} \right\}$$

- This uncertainty can be captured by endowing the weight parameters with the probability distributions $\boldsymbol{\Omega} \sim p(\boldsymbol{\Omega})$ and calculating the posterior distribution of the weights given the training data, $p(\boldsymbol{\Omega}|\mathcal{D})$, where $\mathcal{D} = \{\boldsymbol{\mathcal{X}}^i, \boldsymbol{y}^i\}_{i=1}^{N}$.
- By computing the posterior distribution of the weights given the data, we can find the predictive distribution of any new unseen data point $\boldsymbol{\mathcal{X}}^*$.

$$p(\boldsymbol{y}^*|\boldsymbol{\mathcal{X}}^*, \mathcal{D}) = \int p(\boldsymbol{y}^*|\boldsymbol{\mathcal{X}}^*, \boldsymbol{\Omega})\, p(\boldsymbol{\Omega}|\mathcal{D}) d\boldsymbol{\Omega}$$

Variational Inference (VI) Framework

- The true posterior distribution of the weights given the training data point $p(\boldsymbol{\Omega}\,|\boldsymbol{\mathcal{X}},\boldsymbol{y})$ cannot be evaluated analytically.
- Instead we define an approximating variational distribution $q_\phi(\boldsymbol{\Omega})$, that is easy to evaluate.
- We would like this approximating distribution to be as close as possible to the true unknown posterior distribution.
- We thus minimize the Kullback - Leibler (KL) divergence,

$$KL\left(q_\phi(\boldsymbol{\Omega}) \parallel p(\boldsymbol{\Omega}\,|\boldsymbol{\mathcal{X}},\boldsymbol{y})\right) = -\int q_\phi(\boldsymbol{\Omega}) \log \frac{p(\boldsymbol{\Omega})\, p(\boldsymbol{y}\,|\boldsymbol{\mathcal{X}},\boldsymbol{\Omega})}{p(\boldsymbol{y}\,|\boldsymbol{\mathcal{X}})\, q_\phi(\boldsymbol{\Omega})}\, d\boldsymbol{\Omega}$$

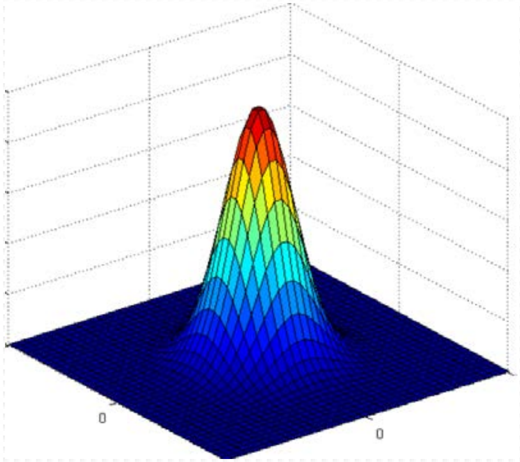
$$= \underbrace{-\int q_\phi(\boldsymbol{\Omega}) \log \frac{p(\boldsymbol{\Omega})\, p(\boldsymbol{y}\,|\boldsymbol{\mathcal{X}},\boldsymbol{\Omega})}{q_\phi(\boldsymbol{\Omega})}\, d\boldsymbol{\Omega}}_{\mathcal{L}(\phi;\,\boldsymbol{y}\,|\boldsymbol{\mathcal{X}})} + \underbrace{\log p(\boldsymbol{y}\,|\boldsymbol{\mathcal{X}})}_{\text{Constant}}$$

(variational) lower bound or evidence lower bound (ELBO) $\mathcal{L}(\phi;\boldsymbol{y}\,|\boldsymbol{\mathcal{X}})$

Constant

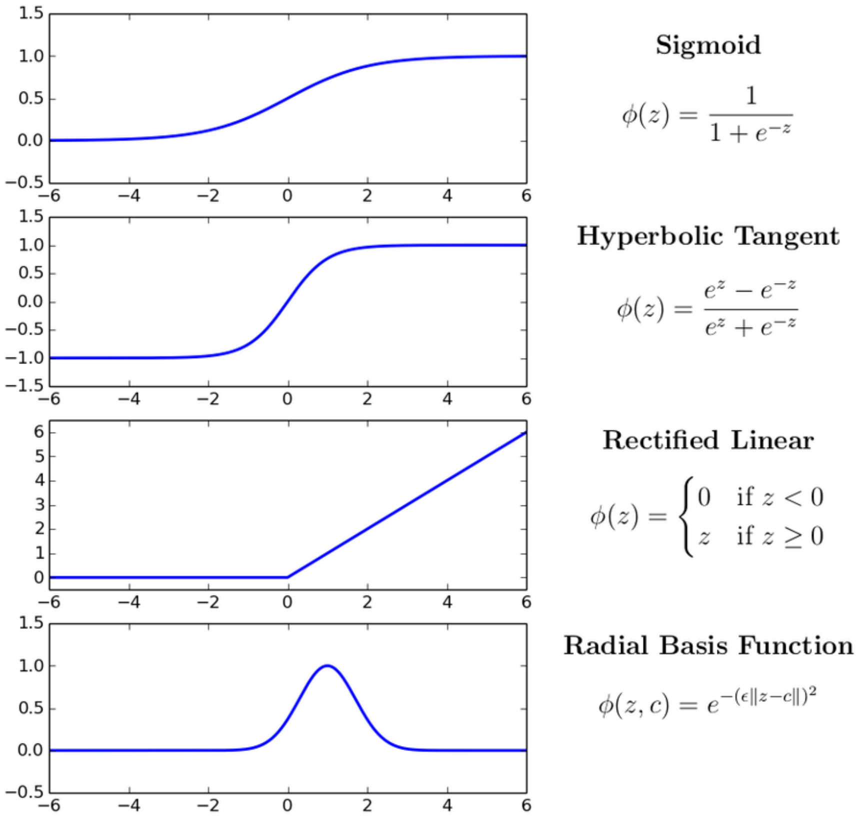
Uncertainty Propagation Challenge

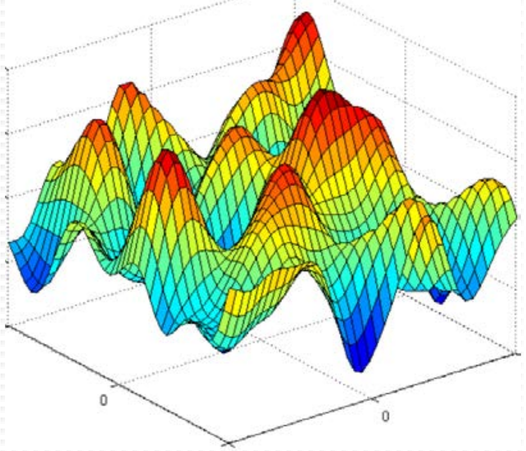
- The challenge remains in propagating the distributions introduced over the weights of the convolutional neural network through multiple layers including the non-linear activation functions.



Multivariate Gaussian Distribution

Nonlinear Activation Functions





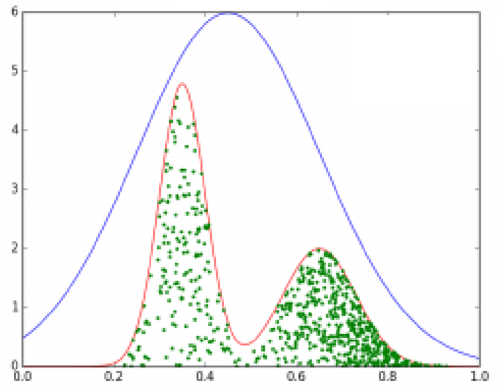
Unknown Distribution

MAIN IDEA

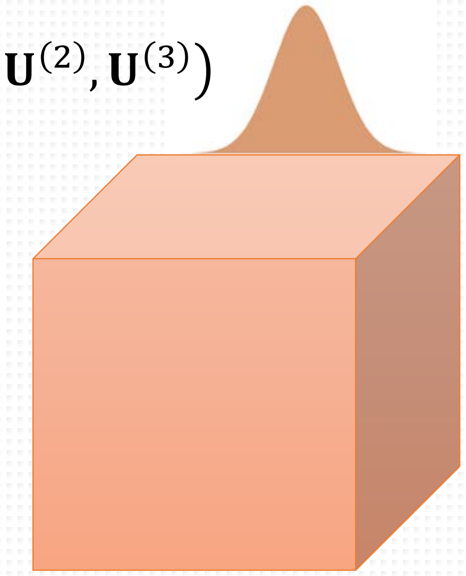
1. Propagate the First Two Moments (Mean and Covariance matrix) across the network using Taylor series approximation.

→ We are basically propagating Gaussians with "good" approximates of true mean and covariances.

2. Propagation of moments through layers of CNN makes it robust to noise (additive, inherent or adversarial) in the data as well as variations in the model parameters (kernels).



Tensor Normal Distribution



- Let $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ be a random 3D tensor. The tensor Normal distribution $\boldsymbol{\mathcal{W}} \sim \mathcal{TN}_{I_1 I_2 I_3}\left(\boldsymbol{\mathcal{M}}; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}\right)$ is defined as,

$$p(\mathcal{W}) = \frac{1}{\sqrt{(2\pi)^{I_1 I_2 I_3}} |\mathrm{U}^{(1)}|^{\frac{I_2 I_3}{2}} |\mathrm{U}^{(2)}|^{\frac{I_1 I_3}{2}} |\mathrm{U}^{(3)}|^{\frac{I_1 I_2}{2}}} \exp\{-\frac{1}{2}(\mathcal{W}-\mathcal{M}) \times_{1\ldots J} (\circ_{j=1}^{3} (\mathrm{U}^{(j)})^{-1}) \times_{1\ldots J} (\mathcal{W}-\mathcal{M})\}$$

where $\boldsymbol{\mathcal{M}}$ is the mean tensor of order 3, and $\mathbf{U}^{(j)}(I_j \times I_j)$, $(j = 1,2,3)$
be positive-definite covariance matrices. [10]

- Equivalent formulation of the tensor Normal distribution is a multivariate Gaussian distribution

$$\boldsymbol{\mathcal{W}} \sim \mathcal{TN}_{I_1, I_2, I_3}\left(\boldsymbol{\mathcal{M}}; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}\right) \Leftrightarrow \mathrm{vec}(\boldsymbol{\mathcal{W}}) \sim \mathcal{N}_{I_1 I_2 I_3}\left(\mathrm{vec}(\boldsymbol{\mathcal{M}});\ \mathbf{U}^{(3)} \otimes \mathbf{U}^{(2)} \otimes \mathbf{U}^{(1)}\right)$$

where vec is the vectorization operation and $\otimes$ is the Kronecker product.

Proposed Extended-Variational Inference - Convolutional Neural Network (EVI-CNN)

- Variational inference framework: Minimize the Kullback-Leibler (KL) divergence between a proposed approximating distribution $q_{\phi}(\boldsymbol{\Omega})$ and the true posterior distribution of the weights. Thus, minimizing the KL-divergence is equivalent to maximizing the ELBO,

$$\mathcal{L}(\phi; \boldsymbol{y} \mid \boldsymbol{\mathcal{X}}) = \int q_{\phi}(\boldsymbol{\Omega}) \log \frac{p(\boldsymbol{\Omega})\, p(\boldsymbol{y} \mid \boldsymbol{\mathcal{X}}, \boldsymbol{\Omega})}{q_{\phi}(\boldsymbol{\Omega})}\, d\boldsymbol{\Omega}$$

$$= E_{q_{\phi}(\boldsymbol{\Omega})}(\log p(\boldsymbol{y} \mid \boldsymbol{\mathcal{X}}, \boldsymbol{\Omega})) - KL(q_{\phi}(\boldsymbol{\Omega}) \parallel p(\boldsymbol{\Omega}))$$

- We assume that the <u>weights of every convolutional or fully connected layers are independent of the weights of other layers.</u>

- We further assume that the <u>kernel tensors in each convolutional layer are independent</u>. This allows us to re-write the ELBO as follows,

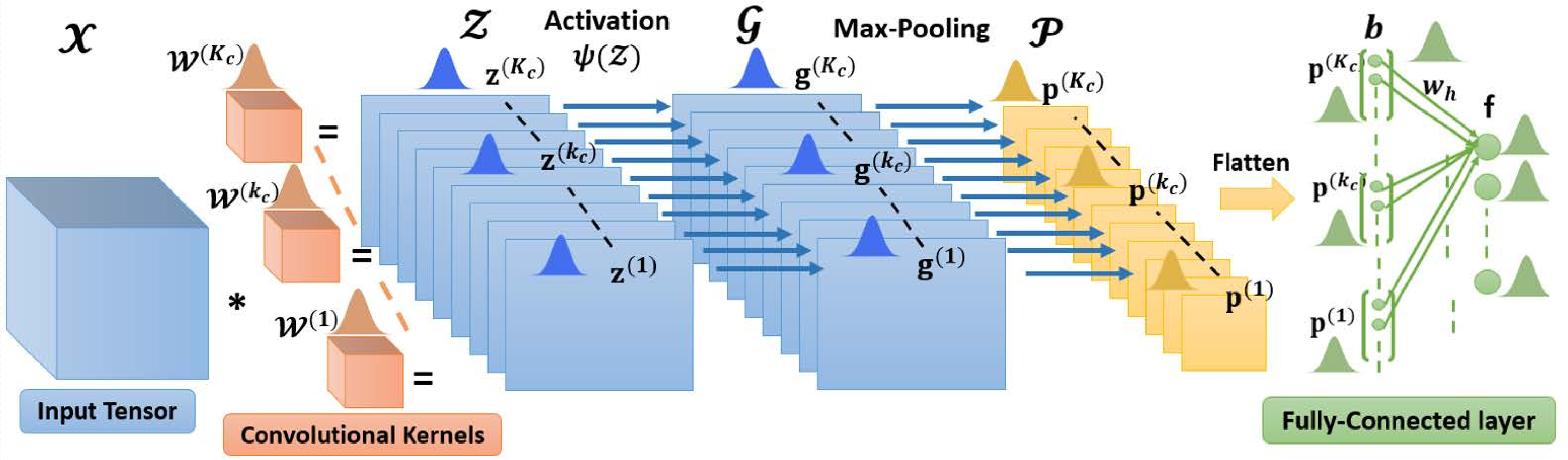
$$\mathcal{L}(\phi; \boldsymbol{y} \mid \boldsymbol{\mathcal{X}}) = E_{\left\{q_{\phi_1}(\boldsymbol{\mathcal{W}}^{(1)}), \dots, q_{\phi_{K_c}}(\boldsymbol{\mathcal{W}}^{(K_c)})\right\}_{c=1}^{C}, q_{\phi_1}(\mathbf{W}^{(1)}), \dots, q_{\phi_L}(\mathbf{W}^{(L)})} \left(\log p\left(\boldsymbol{y} \mid \boldsymbol{\mathcal{X}}, \{\boldsymbol{\mathcal{W}}^{(1)}, \dots, \boldsymbol{\mathcal{W}}^{(K_c)}\}_{c=1}^{C}, \mathbf{W}^{(1)}, \dots, \mathbf{W}^{(L)}\right)\right)$$

$$- \sum_{c=1}^{C} \sum_{k_c=1}^{K_c} KL\left(q_{\phi_{k_c}}(\boldsymbol{\mathcal{W}}^{(k_c)}) \parallel p(\boldsymbol{\mathcal{W}}^{(k_c)})\right) - \sum_{l=1}^{L} KL\left(q_{\phi_l}(\mathbf{W}^{(l)}) \parallel p(\mathbf{W}^{(l)})\right)$$

Uncertainty Propagation Across CNN Layers - Derivation of the Lower Bound

We consider a convolutional neural network with:

- ❑ One convolutional layer
- ❑ Nonlinearity (e.g. ReLU activation),
- ❑ Max-pooling layer,
- ❑ One fully connected.



Derivation of the Expected Log-Likelihood

- We use Monte Carlo simulation to approximate the expectation of the log-likelihood by a summation by sampling the weights of every convolutional and fully-connected layers from the approximate distribution $q_\phi(\boldsymbol{\Omega})$.

$$E_{q_\phi(\boldsymbol{\Omega})}(\log p(\boldsymbol{y}\,|\boldsymbol{\mathcal{X}},\boldsymbol{\Omega})) \approx \frac{1}{M}\sum_{m=1}^{M}\log p\left(\boldsymbol{y}\,|\boldsymbol{\mathcal{X}},\boldsymbol{\Omega}^{(m)}\right)$$

- The distribution $p(\boldsymbol{y}\,|\boldsymbol{\mathcal{X}},\boldsymbol{\Omega})$ is assumed multivariate Gaussian. Its mean and covariance matrix are estimated by propagating the mean and covariance of the approximating variational distribution $q_\phi(\boldsymbol{\Omega})$ through the network.

Propagation of Mean and Covariance

1. At the output of convolution:

$$\mathbf{z}^{(k_c)} \sim \mathcal{N}\left(\tilde{\mathbf{X}}\mathbf{m}^{(k_c)},\ \tilde{\mathbf{X}}\boldsymbol{\Sigma}^{(k_c)}\tilde{\mathbf{X}}^T\right)$$

2. At the output of the non-linear activation layer - first order Taylor series linearization:

$$\mathrm{E}[\mathbf{g}_i^{(k_c)}] \approx \psi(\mathrm{E}[\mathbf{z}_i^{(k_c)}]);$$

$$\mathrm{Var}[\mathbf{g}_i^{(k_c)}] \approx \sigma^2_{\mathbf{z}_i^{(k_c)}} \left(\frac{d\psi(\mu_{\mathbf{z}_i^{(k_c)}})}{d\mathbf{z}_i^{(k_c)}}\right)^2;$$

$$\mathrm{Cov}[\mathbf{g}_i^{(k_c)}, \mathbf{g}_j^{(k_c)}] \approx$$

$$\approx \sigma_{\mathbf{z}_i^{(k_c)}\mathbf{z}_j^{(k_c)}} \left(\frac{d\psi(\mu_{\mathbf{z}_i^{(k_c)}})}{d\mathbf{z}_i^{(k_c)}}\right)\left(\frac{d\psi(\mu_{\mathbf{z}_j^{(k_c)}})}{d\mathbf{z}_j^{(k_c)}}\right), i \neq j.$$

Propagation of Mean and Covariance

3. At the output of the pooling layer – downsample:

$$\boldsymbol{\mu}_{\mathbf{p}^{(k_c)}} = \text{pool}(\boldsymbol{\mu}_{\mathbf{g}^{(k_c)}}) \qquad \boldsymbol{\Sigma}_{\mathbf{p}^{(k_c)}} = \text{pool}(\boldsymbol{\Sigma}_{\mathbf{g}^{(k_c)}})$$

4. At the output of the fully connected layer:

$$E[\mathbf{f}_h] = \mathbf{m}_h^T \boldsymbol{\mu}_{\mathbf{b}};$$
$$Var[\mathbf{f}_h] = \text{tr}\left(\boldsymbol{\Sigma}_h \boldsymbol{\Sigma}_{\mathbf{b}}\right) + \mathbf{m}_h^T \boldsymbol{\Sigma}_{\mathbf{b}} \mathbf{m}_h + \boldsymbol{\mu}_{\mathbf{b}}^T \boldsymbol{\Sigma}_h \boldsymbol{\mu}_{\mathbf{b}};$$
$$Cov[\mathbf{f}_{h_i}, \mathbf{f}_{h_j}] = \mathbf{m}_{h_i}^T \boldsymbol{\Sigma}_{\mathbf{b}} \mathbf{m}_{h_j}, i \neq j.$$

BACKPROPAGATION

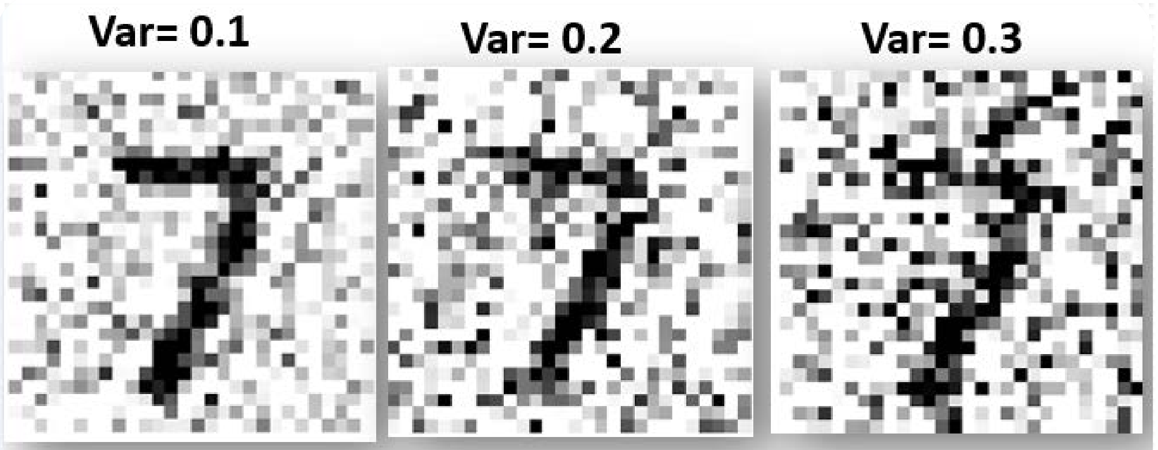
- In the forward pass, we propagated the mean and covariances of the approximate distributions $q_{\phi}(\boldsymbol{\Omega})$ across the network layers and calculated the objective function.

- In the back-propagation pass, we compute the gradient of the objective function w.r.t the variational parameters and update in an iterative procedure.

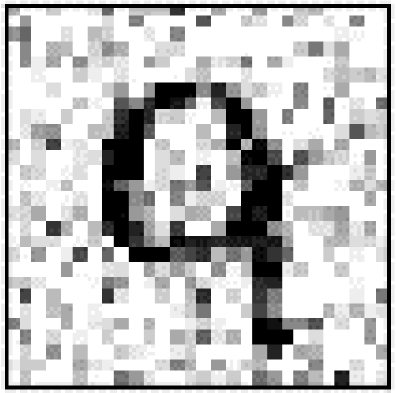
Simulation Results and Discussion

Robustness to Gaussian Noise on MNIST Dataset

Gaussian Noise			
0.1	94%	86%	79%
0.2	85%	76%	70%
0.3	**74%**	**63%**	**55%**
Zero (No noise)	96%	96%	96%



Output Mean and Covariance Matrix for Input Corrupted with 0.1 Gaussian Noise (Correctly Classified Input)



True: 9, Pred: 9

Ground Truth (yellow)
Network Prediction (green)

If the yellow block is not shown, then the network prediction and the ground truth are the same.

Output Covariance Matrix

	0	1	2	3	4	5	6	7	8	9
0	0.0133	8E-07	0.0002	2E-05	0.0003	2E-05	3E-05	0.0001	0.00017	0.0029
1	8E-07	7E-08	1E-07	4E-08	5E-07	5E-08	6E-08	3E-07	2.99E-07	5E-06
2	0.0002	1E-07	0.0155	1E-05	0.0003	9E-07	2E-05	4E-05	0.000109	0.0013
3	2E-05	4E-08	1E-05	8E-05	6E-06	2E-06	1E-06	9E-06	7.25E-06	0.0002
4	0.0003	5E-07	0.0003	6E-06	0.1216	-2E-05	2E-05	0.0001	0.000373	0.0022
5	2E-05	5E-08	9E-07	2E-06	-2E-05	1E-04	8E-07	1E-05	8.07E-06	0.0003
6	3E-05	6E-08	2E-05	1E-06	2E-05	8E-07	0.0002	4E-06	1.33E-05	0.0001
7	0.0001	3E-07	4E-05	9E-06	0.0001	1E-05	4E-06	0.004	6.16E-05	0.0016
8	0.0002	3E-07	0.0001	7E-06	0.0004	8E-06	1E-05	6E-05	0.0034	0.0018
9	0.0029	5E-06	0.0013	0.0002	0.0022	0.0003	0.0001	0.0016	0.001753	0.5083

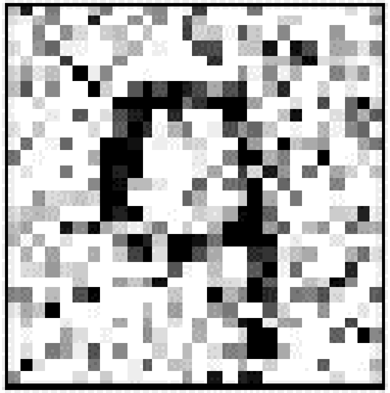
Output Prediction

0	
0.005	0
1E-05	1
0.0056	2
0.0004	3
0.0153	4
0.0004	5
0.0005	6
0.0028	7
0.0026	8
0.9674	9

Prediction of Deterministic CNN

0.0009	0
6E-06	1
0.0019	2
0.0007	3
0.0234	4
0.0121	5
0.0006	6
0.0038	7
0.0037	8
0.9529	9

Output Mean and Covariance Matrix for Input Corrupted with 0.2 Gaussian Noise (Correctly Classified Input)



True: 9, Pred: 9

Ground Truth (yellow)
Network Prediction (green)

If the yellow block is not shown, then the network prediction and the ground truth are the same.

Output Covariance Matrix

	0	1	2	3	4	5	6	7	8	9
0	0.5692	1.1E-05	0.00889	0.00035	0.00055	0.00468	0.00194	0.00262	0.00228	0.04001
1	1.1E-05	1.6E-06	1.1E-05	8.1E-07	-3E-08	1.6E-05	4.7E-06	5.9E-07	3.3E-06	8.3E-05
2	0.00889	1.1E-05	0.74749	0.00061	0.00153	-0.0035	0.00355	0.0035	0.00227	0.03168
3	0.00035	8.1E-07	0.00061	0.00425	-6E-06	0.00051	0.00018	0.00019	6.6E-05	0.00418
4	0.00055	-3E-08	0.00153	-6E-06	0.06475	-0.002	0.00018	0.00098	0.00022	0.00179
5	0.00468	1.6E-05	-0.0035	0.00051	-0.002	0.92187	0.00145	0.00145	0.0006	0.07743
6	0.00194	4.7E-06	0.00355	0.00018	0.00018	0.00145	0.05348	1.2E-05	0.00034	0.01141
7	0.00262	5.9E-07	0.0035	0.00019	0.00098	0.00145	1.2E-05	0.18983	0.0016	0.02907
8	0.00228	3.3E-06	0.00227	6.6E-05	0.00022	0.0006	0.00034	0.0016	0.05928	0.01495
9	0.04001	8.3E-05	0.03168	0.00418	0.00179	0.07743	0.01141	0.02907	0.01495	10.8337

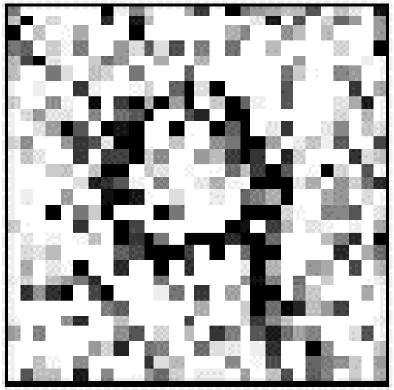
Output Prediction

0	
0.0309	0
5E-05	1
0.036	2
0.0027	3
0.0102	4
0.0397	5
0.0092	6
0.0177	7
0.0101	8
0.8435	9

Prediction of Deterministic CNN

0.00185	0
2E-05	1
0.0016	2
0.00516	3
0.01037	4
0.02767	5
0.00159	6
0.03252	7
0.02757	8
0.89166	9

Output Mean and Covariance Matrix for Input Corrupted with 0.3 Gaussian Noise (Correctly Classified Input)



True: 9, Pred: 9

Ground Truth (yellow)

Network Prediction (green)

If the yellow block is not shown, then the network prediction and the ground truth are the same.

Output Covariance Matrix

	0	1	2	3	4	5	6	7	8	9
0	0.72614	7.3E-06	0.02063	0.00011	0.00301	0.00041	5.4E-05	0.02911	0.00164	0.10533
1	7.3E-06	1.4E-06	5.3E-06	4.6E-08	4.9E-07	4.9E-07	2.9E-08	1.4E-05	1.2E-06	4.7E-05
2	0.02063	5.3E-06	6.30839	0.00028	0.00323	-2E-05	0.0001	0.04422	0.00198	0.1513
3	0.00011	4.6E-08	0.00028	0.00012	2.7E-06	6E-06	3.2E-07	0.00017	5.1E-06	0.00115
4	0.00301	4.9E-07	0.00323	2.7E-06	0.05826	-0.0001	9.8E-06	0.00286	0.00048	0.01931
5	0.00041	4.9E-07	-2E-05	6E-06	-0.0001	0.00451	1.1E-06	0.00222	1.7E-05	0.00518
6	5.4E-05	2.9E-08	0.0001	3.2E-07	9.8E-06	1.1E-06	1.2E-05	7E-05	5.2E-06	0.00024
7	0.02911	1.4E-05	0.04422	0.00017	0.00286	0.00222	7E-05	10.2073	0.00447	0.2822
8	0.00164	1.2E-06	0.00198	5.1E-06	0.00048	1.7E-05	5.2E-06	0.00447	0.00649	0.01209
9	0.10533	4.7E-05	0.1513	0.00115	0.01931	0.00518	0.00024	0.2822	0.01209	29.1693

Output Prediction

0	
0.03217	0
4.3E-05	1
0.10437	2
0.00042	3
0.00889	4
0.00248	5
0.00013	6
0.1372	7
0.00301	8
0.71129	9

Prediction of Deterministic CNN

1.1E-11	0
3.9E-16	1
0.00083	2
3.1E-08	3
0.02638	4
0.00825	5
3.2E-18	6
8.5E-06	7
3.8E-06	8
0.96454	9

Output Mean and Covariance Matrix for Input Corrupted with 0.4 Gaussian Noise (Correctly Classified Input)

True: 9, Pred: 9

True: 9, Pred: 4

Ground Truth (yellow)
Network Prediction (green)

If the yellow block is not shown, then the network prediction and the ground truth are the same.

Output Covariance Matrix

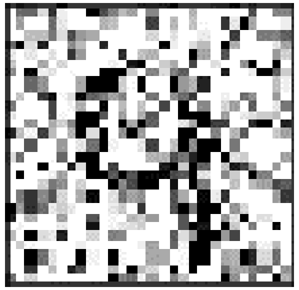
	0	1	2	3	4	5	6	7	8	9
0	0.35451	0.00243	0.33682	0.10871	0.02829	0.31088	0.00945	0.11341	0.07531	-1.33981
1	0.00243	0.00025	0.00921	0.00302	0.00079	0.0101	0.00026	0.00312	0.00205	-0.03122
2	0.33682	0.00921	8.09456	0.41646	0.10979	0.36971	0.03833	0.45531	0.28008	-10.1103
3	0.10871	0.00302	0.41646	0.56804	0.03578	0.3718	0.0117	0.14765	0.09329	-1.75646
4	0.02829	0.00079	0.10979	0.03578	0.03635	0.11465	0.00306	0.03919	0.02511	-0.39302
5	0.31088	0.0101	0.36971	0.3718	0.11465	32.0097	0.03896	0.41098	0.26386	-33.9007
6	0.00945	0.00026	0.03833	0.0117	0.00306	0.03896	0.0038	0.01245	0.00801	-0.12601
7	0.11341	0.00312	0.45531	0.14765	0.03919	0.41098	0.01245	0.66907	0.09918	-1.95036
8	0.07531	0.00205	0.28008	0.09329	0.02511	0.26386	0.00801	0.09918	0.25456	-1.10146
9	-1.33981	-0.03122	-10.1103	-1.75646	-0.39302	-33.9007	-0.12601	-1.95036	-1.10146	50.7092

Output Prediction

0	
0.0206	0
0.0005	1
0.1057	2
0.0268	3
0.0065	4
0.2419	5
0.0021	6
0.0289	7
0.0178	8
0.5491	9

Prediction of Deterministic CNN (misclassified)

3.2E-24	0
1.4E-13	1
5.1E-13	2
4.3E-18	3
0.93953	4
1.9E-19	5
3.8E-32	6
0.06046	7
6.3E-06	8
2.1E-14	9



True: 9, Pred: 2

Output Mean and Covariance Matrix for Input Corrupted with 0.5 Gaussian Noise (Misclassified Input)

True: 9, Pred: 3

Yellow: Ground Truth
Green: Network Prediction

If the yellow block is not shown, then the network prediction and the ground truth are the same.

Output Covariance Matrix

	0	1	2	3	4	5	6	7	8	9
0	0.0556	0.00065	-0.2905	0.03488	0.13344	0.13218	0.01133	0.02721	0.02012	-0.1249
1	0.00065	0.00016	-0.0157	0.0019	0.00759	0.00771	0.00062	0.00149	0.00108	-0.0055
2	-0.2905	-0.0157	98.5714	-0.9234	-9.3617	-16.419	-0.2636	-0.663	-0.5908	-70.043
3	0.03488	0.0019	-0.9234	0.55612	0.40877	0.35715	0.03283	0.08706	0.05913	-0.6145
4	0.13344	0.00759	-9.3617	0.40877	15.848	0.38258	0.13139	0.3363	0.22686	-8.1132
5	0.13218	0.00771	-16.419	0.35715	0.38258	23.9703	0.12738	0.28712	0.22527	-9.0704
6	0.01133	0.00062	-0.2636	0.03283	0.13139	0.12738	0.05154	0.02534	0.01901	-0.1359
7	0.02721	0.00149	-0.663	0.08706	0.3363	0.28712	0.02534	0.31552	0.04433	-0.4613
8	0.02012	0.00108	-0.5908	0.05913	0.22686	0.22527	0.01901	0.04433	0.16738	-0.1724
9	-0.1249	-0.0055	-70.043	-0.6145	-8.1132	-9.0704	-0.1359	-0.4613	-0.1724	88.7415

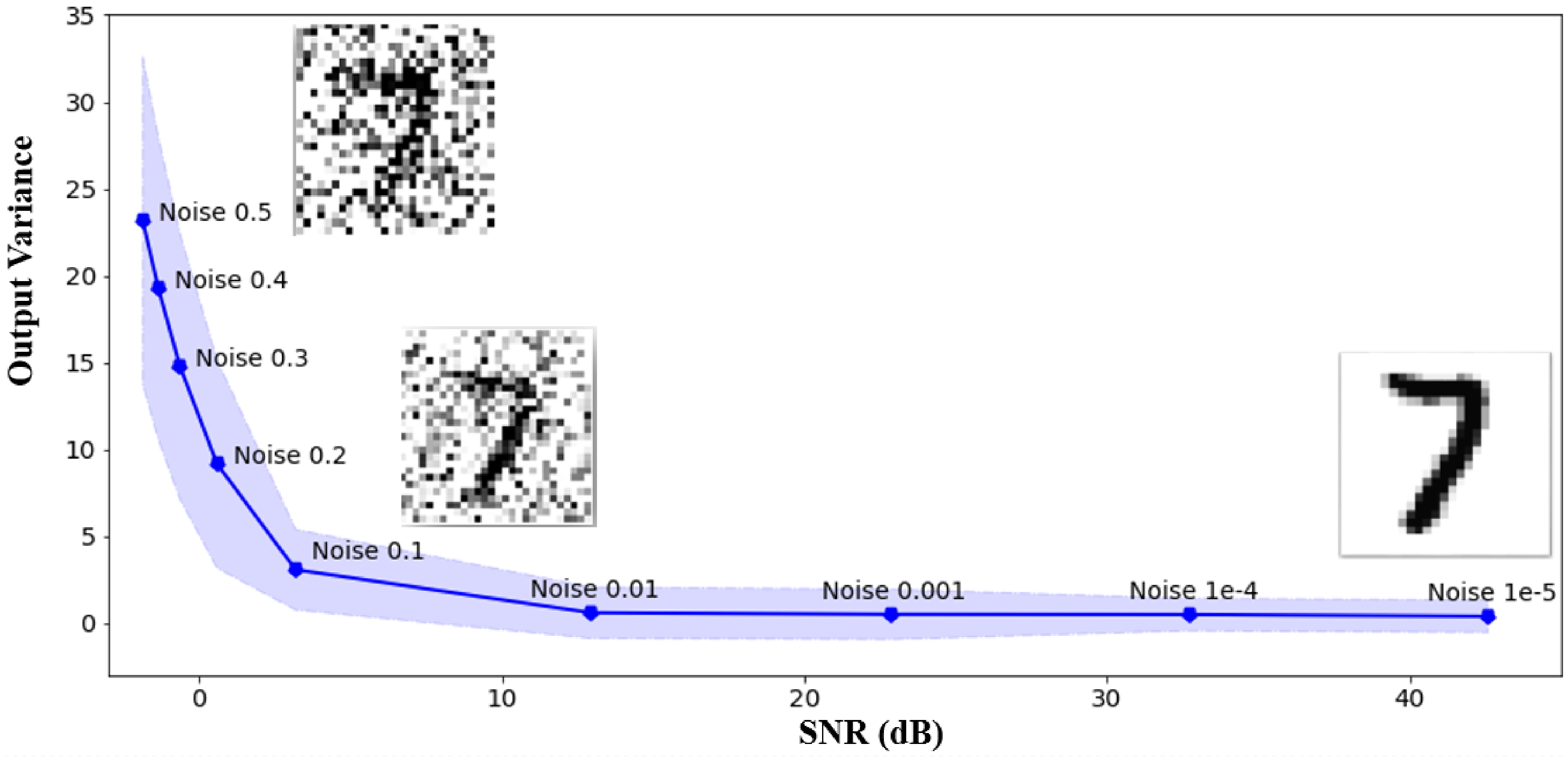
Output Prediction

0	
0.00596	0
0.00032	1
0.37629	2
0.01879	3
0.10606	4
0.13716	5
0.00569	6
0.01405	7
0.01038	8
0.32531	9

Prediction of Deterministic CNN (misclassified)

6.1E-30	0
8.3E-24	1
5.8E-08	2
0.98983	3
1.4E-15	4
0.00946	5
0	6
0.0007	7
1.2E-17	8
1.3E-28	9

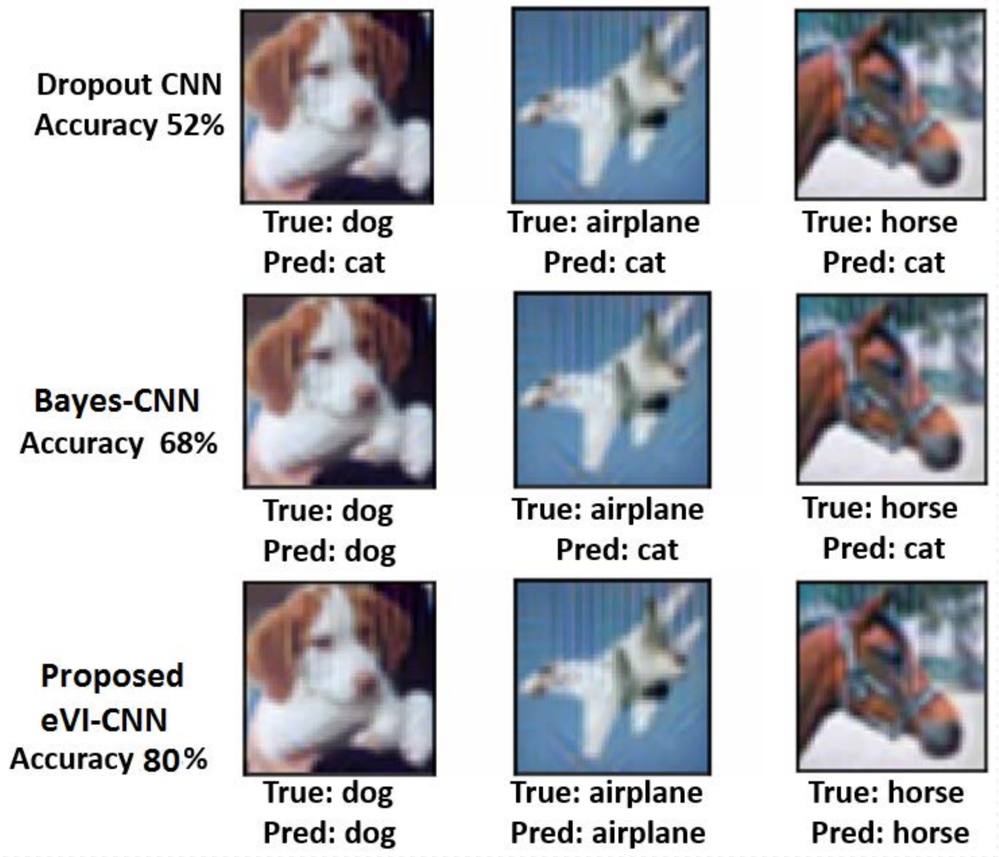
THE VARIANCE OF THE OUTPUT PREDICTION OF EXTENDED-VI VS. SIGNAL-TO-NOISE RATIO (SNR).



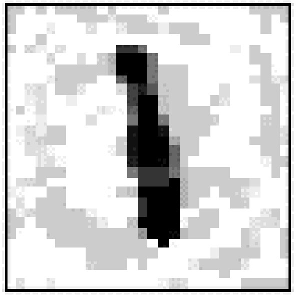
ROBUSTNESS TO ADVERSARIAL ATTACKS

- Adversarial attack: Generate some image that is designed to make the network have a certain output. For instance, say our goal label/output is y_{goal}. Let's call the attack image we want to generate X. X is found by optimizing

$$C(X) = \frac{1}{2} \left\| y_{goal} - \hat{y}(X + \text{training}) \right\|_2^2$$



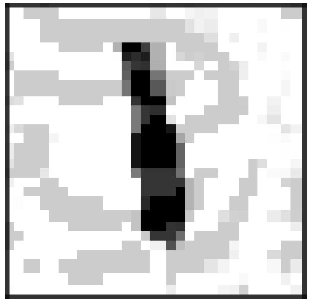
Prediction of the Dropout network, Bayes-CNN and eVI-CNN for three randomly chosen images from the CIFAR-10 dataset corrupted by an adversarial noise created to fool each network into predicting the class label as a "cat". The adversarial noise was created at the same level, i.e. 5% for all networks.



True: 1, Pred: 3

Output of Softwmax is not a probability

Example: 0.2 Adversarial Noise (Misclassified Input)



True: 1, Pred: 3

- Yellow: Ground Truth
- Green: Network Prediction

If the yellow block is not shown, then the network prediction and the ground truth are the same.

Output Covariance Matrix

	0	1	2	3	4	5	6	7	8	9
0	2.8E-12	2E-07	1.1E-09	1.4E-07	2.3E-10	8E-11	4.2E-11	1.1E-09	1.8E-09	2.1E-11
1	2E-07	1.39853	0.00093	0.11747	0.0003	8.4E-05	4.7E-05	0.0006	0.00154	2E-05
2	1.1E-09	0.00093	7.9E-05	0.00079	1.3E-06	3.5E-07	2.9E-07	7.7E-07	6.6E-06	1.1E-07
3	1.4E-07	0.11747	0.00079	1.09673	0.00017	6.9E-05	3.8E-05	0.00068	0.00088	1.4E-05
4	2.3E-10	0.0003	1.3E-06	0.00017	4.4E-06	1.1E-07	1.1E-07	1.3E-06	2.2E-06	2.9E-08
5	8E-11	8.4E-05	3.5E-07	6.9E-05	1.1E-07	5.7E-07	1.8E-08	1.1E-07	7.6E-07	8.7E-09
6	4.2E-11	4.7E-05	2.9E-07	3.8E-05	1.1E-07	1.8E-08	1E-07	1.8E-07	3.7E-07	4.5E-09
7	1.1E-09	0.0006	7.7E-07	0.00068	1.3E-06	1.1E-07	1.8E-07	4.4E-05	6.9E-06	1E-07
8	1.8E-09	0.00154	6.6E-06	0.00088	2.2E-06	7.6E-07	3.7E-07	6.9E-06	0.00018	2E-07
9	2.1E-11	2E-05	1.1E-07	1.4E-05	2.9E-08	8.7E-09	4.5E-09	1E-07	2E-07	1.7E-08

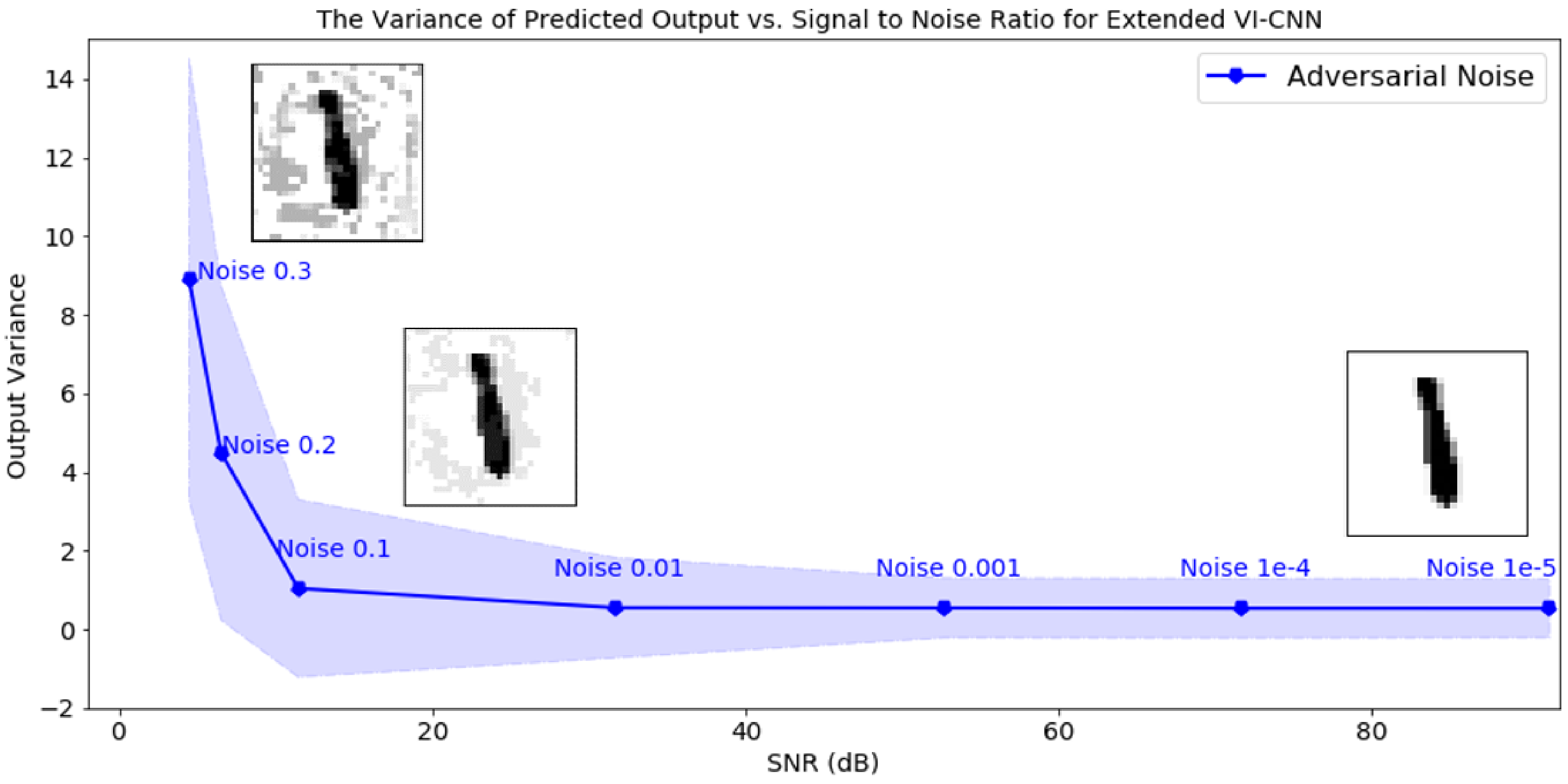
Output Prediction

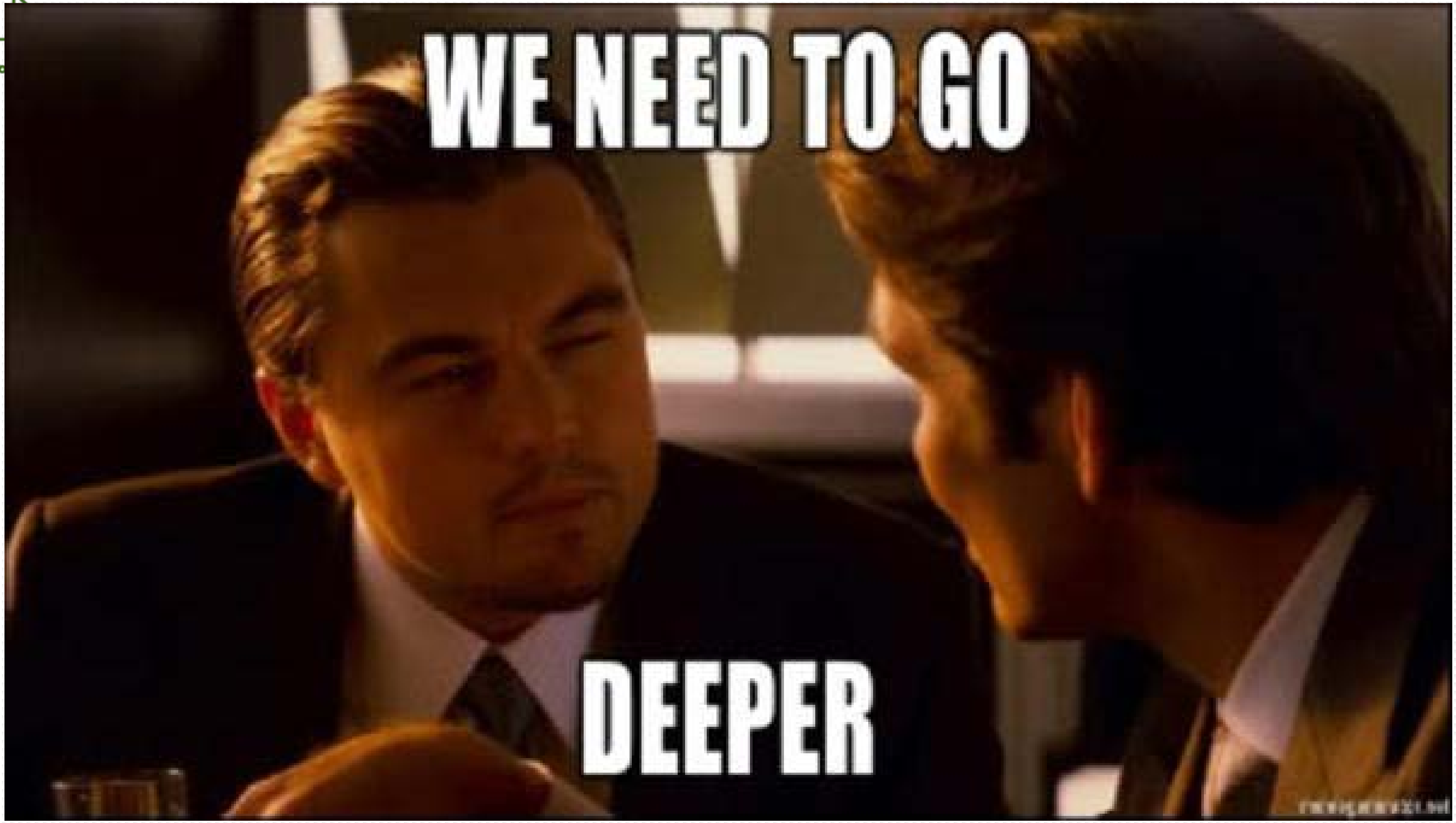
2E-07	0
0.1529	1
0.0011	2
0.8427	3
0.0002	4
9E-05	5
3E-05	6
0.0011	7
0.0018	8
2E-05	9

Prediction of Deterministic CNN

2E-13	0
0.0238	1
8E-08	2
0.9762	3
7E-10	4
5E-08	5
5E-10	6
1E-09	7
8E-06	8
2E-13	9

The Variance of the Output Prediction of Extended-VI vs. Signal-to-Noise Ratio (SNR)

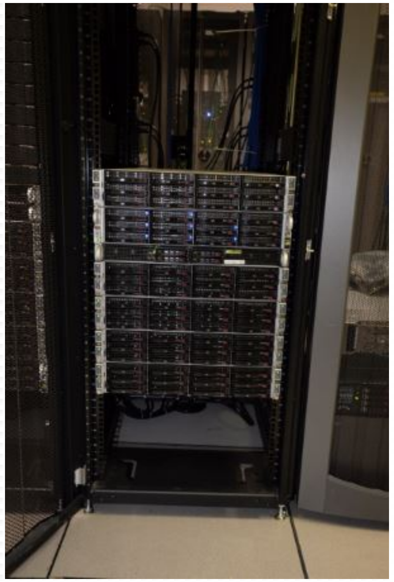



WE NEED TO GO
DEEPER

THANKS TO …

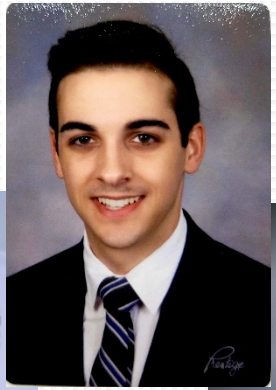

Hassan M. Fathallah-Shaykh


Roman Shterenberg




Ghulam Rasool

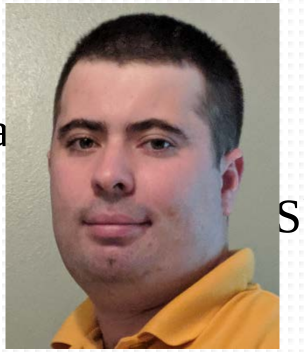

Daniel Ca


Jacob Epifan


Dimah Dera


Gad Souissi


Chris Angelini


David Specht


Shamoon Siddiqui


Hikmat Khan


Giuseppina Carannante


Eric Feuerstein