Kernelized Capsule Networks

Taylor Killian [* 1] **Justin Goodwin** [* 1] **Olivia Brown** [1] **Sung-Hyun Son** [1]

Abstract

Capsule Networks attempt to represent patterns in images in a way that preserves hierarchical spatial relationships. Additionally, research has demonstrated that these techniques may be robust against adversarial perturbations. We present an improvement to training capsule networks with added robustness via non-parametric kernel methods. The representations learned through the capsule network are used to construct covariance kernels for Gaussian processes (GPs). We demonstrate that this approach achieves comparable prediction performance to Capsule Networks while improving robustness to adversarial perturbations and providing a meaningful measure of uncertainty that may aid in the detection of adversarial inputs.

1. Introduction

Neural networks are a powerful computational paradigm capable of discovering expressive representations of nearly any modality of data. Representation learning with neural networks is provided through optimizing multiple layers of adaptive basis functions. Novel and creative research has introduced highly specialized functions and network architectures applied across a variety of scientific disciplines and commercial industries. However, traditional neural network models, once trained, are deterministic and prone to decaying performance as data distributions shift (Heckman, 1979; Sugiyama et al., 2007) or adversarial inputs are introduced (Madry et al., 2017; Szegedy et al., 2013). To overcome these various weaknesses, researchers have developed alternative formulations for neural architectures.

One such neural architecture is the Capsule Network (Hin-

ton et al., 2011). Introduced in 2011, Capsule Networks (CapsNets) replace the scalar "neurons" with "capsules," activity vectors that encapsulate the probability of a visual entity being present along with information about the entity's parameters (e.g., pose). Capsules are expected to provide general viewpoint equivariance across spatial transformations. CapsNets have shown promising results in recent research (Hinton et al., 2018; Kumar, 2018; Sabour et al., 2017) on simple image classification problems using a multi-class margin loss. Additionally, these networks have been shown to have some robustness against adversarial perturbations (Frosst et al., 2018; Hinton et al., 2018). Yet, Marchisio et al. (2019) demonstrate that robustness to adversarial perturbations is not guaranteed with existing architectures.

We aim to improve the robustness of CapsNets by offering an alternative training approach, making use of a Gaussian Process (GP) formed through the construction of a covariance kernel from the capsule output. In addition to the new training approach, the GP classification layer provides a mechanism for detecting adversarial or otherwise "strange" inputs to the model through the entropy of the predictive posterior distribution (Bradshaw et al., 2017). This signal may possibly be utilized in conjunction with the reconstruction approach introduced by Frosst et al. (2018) to identify inputs that the network is uncertain about. Most importantly, we show that a Kernelized Capsule Network (KCN) is more robust to adversarial perturbations while also capable of detecting examples that lie outside it's learned data distribution.

2. Background

Capsule Networks Sabour et al. (2017) introduced a dynamic routing by agreement approach to building CapsNets. The hierarchical feature representations are iteratively refined by calculating the agreement between the proposal and input capsules. The activation is simply the length of a given capsule and corresponds to the probability that an entity represented by the capsule is present. The CapsNet parameters are optimized using a margin loss for each output class. Additionally, a regularization method is applied via an $\ell 2$ reconstruction loss to encourage each capsule to encode parameters of the input image. In our KCN formula-

tion, we utilize this same reconstruction loss, but provide an alternative to the margin loss through variational inference.

Deep Kernel Learning   Deep Kernel Learning (Wilson et al., 2016) is an innovative hybrid GP and deep neural network (DNN) approach that utilizes mini-batch training to learn the GP kernel for supervised learning tasks such as image classification. Using a neural network to transform the high dimensional pixel data to a set of local features, $v = \mathcal{I}(x)$, a Gaussian Processes model is applied to the set of features via variational learning with inducing points. There are a number of approaches used to solve sparse GP models, e.g., kernel interpolation via KISS-GP (Wilson & Nickisch, 2015). Instead of kernel interpolation techniques, we take advantage of the variational learning methods introduced in (Titsias, 2009; Hensman et al., 2015).

Adversarial perturbations   Given a trained network, $\mathcal{I}(\cdot\,;\theta)$, an adversarial perturbation to an input, x, is a small perturbation, η, that causes the perturbed input to be misclassified, i.e., $\mathcal{I}(x + \eta;\theta) \neq \mathcal{I}(x;\theta)$. Existing methods for computing adversarial perturbations (often referred to as "adversarial attacks") range from quick approximations that take only a single gradient step, such as the "fast gradient sign method" (FGSM) (Goodfellow et al., 2014), to solving the full optimization as in Carlini & Wagner (2017). FGSM is useful for evaluating a network's robustness by varying the perturbation size, ϵ, and perturbations are computed as $\eta = \epsilon\,\text{sign}(\nabla_x\,\mathcal{L}(x,y))$, where $\mathcal{L}$ is the loss function used to find adversarial examples. For white box attacks, we use the loss function for our given network, $L_{\mathcal{I}}(x,y;\theta)$, and for black box attacks, we use the loss function for a surrogate model, $L_{\mathcal{I}'}(x,y;\theta)$.

3. Kernelized Capsule Networks

We introduce the Kernelized Capsule Network (KCN), a hybrid GP and DNN model that brings together CapsNets and Deep Kernel Learning (DKL). The intent of combining the meaningful representations learned through the CapsNet and the flexible expressivity of GP kernel functions is to more robustly train the CapsNet via the marginal likelihood of the kernel function. Beyond robustness, the KCN has the intrinsic capability of signalling, through the posterior distribution of the induced GP, when the model is uncertain and the predictions it provides are not reliable.

A KCN (Figure 1) is built from the nominal CapsNet architecture (Sabour et al., 2017), with two convolutional input layers at the head of the primary capsule layer. Intermediate feature representations from the primary capsules are converted into output capsules, $\mathcal{V} = \{v_1, ...v_{N_c} | v_j \in \mathbb{R}^{k \times 1}\}$, through the dynamic routing procedure. We treat the set of output capsules, $\mathcal{I}(x) = \mathcal{V}$, as a final feature representation
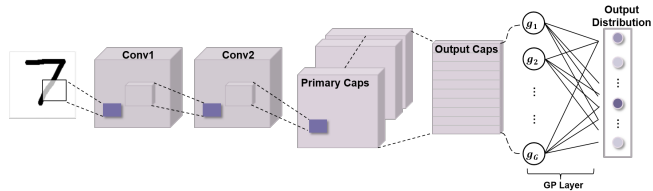

Figure 1. The Kernelized Capsule Network

that is used to construct a GP kernel function (Section 3.1), where the correlation of individual features are measured. This induced GP is then used to infer the probability of the input image's label by way of a Softmax likelihood (eqn. 4).

3.1. Variational Learning

The challenge to learning with capsule networks is providing a meaningful loss when given a vector instead of the standard logits of neural networks. Given capsules $v_j \in \mathcal{V}$, we want to construct a loss that helps capture the learning goal of capsule networks (equivariance to spatial transformation) while accounting for the correlations between features and providing robustness to adversarial perturbations. To achieve this goal we utilize Gaussian Processes, namely Deep Kernel Learning. The input data will be the set of capsules $\mathcal{V}$ with the goal of estimating the class probabilities.

To address the scalability of GPs, we use stochastic variational inference to approximate the matrix inversion and factorization operations used in calculating the predictive posterior distribution (Titsias, 2009; Wilson & Nickisch, 2015). First, define the kernel function $K_{xz} = k(X, Z)$ where,

$$k(X, Z) = \exp(-\gamma\|\mathcal{I}(X) - \mathcal{I}(Z)\|^2), \qquad (1)$$

and γ is a free parameter. Then, as shown in Hensman et al. (2015), the approximate posterior Gaussian Process is defined by $q(f) = \mathcal{N}(f|\mu, \Sigma)$ where the mean is

$$\mu = k(X, Z)k(Z, Z)^{-1}m = K_{xz}K_{zz}^{-1}m \qquad (2)$$

and the covariance is

$$\Sigma = K_{xx} - K_{xz}(K_{zz}^{-1} - K_{zz}SK_{zz}^{-1})K_{zx} \qquad (3)$$

where the m and S are the variational parameters of the distribution $q(u) = \mathcal{N}(u|m, S)$, while Z represents the learned inducing variables. The parameters are optimized by maximizing the bound on the marginal log likelihood for N data points,

$$log(p(y)) \geq \sum_{n=1}^{N} \mathbb{E}_{q(f_n)}[\log(p(y_n|f_n)) - KL[q(u)\|p(u)] \quad (4)$$

This is a summation across the marginal probabilities because we can factor the likelihood for classification as

$$p(\mathbf{y}|f, X) = \prod_n p(y_n|f(x_n)) \qquad (5)$$

For our classification problem we utilize the Softmax likelihood using the categorical distribution.

3.2. Model Usage

With a fully trained KCN, images are first input to the CapsNet. The capsule representations, $\mathcal{V}$, are then passed through the kernel, which via eqt. 2 and eqt. 3 construct a set of correlated GPs. The latent functions produced by these GPs are then passed through the marginal log likelihood (eqt. 4) to estimate the classification probabilities. Finally, predictions are made by taking the $argmax$ of that output distribution, with the expectation that the highest probability corresponds to the correct class of the input image.

3.3. Detection of Adversarial Inputs

Frosst et al. (2018) demonstrated that an auxiliary decoder network, $d(I(x))$, could be trained to reconstruct the input image from the capsule features corresponding to the true class $\{v_j \mid j = y\}$ (see Figure 5 in the appendix). This enables the detection of adversarially perturbed inputs, as perturbations corrupt the features of the capsule representation, affecting the $\ell 2$ accuracy of the reconstruction when compared to the original input. Bradshaw et al. (2017) showed that an analogous mechanism could be developed for a standard deep neural network (DNN) by measuring the entropy of the posterior predictive distribution trained through a hybrid GP-DNN. Motivated by these two papers, we seek to investigate how each signal can contribute to the detection of adversarial inputs.

To detect adversarial inputs, we assume "normal" network behavior when the reconstruction error is less than or equal a given threshold and perturbed images are detected when the threshold is exceeded. We examine Receiver Operating Characteristic (ROC) curves to understand the detection performance. Similarly, we investigate using a threshold on the entropy of the network's predictions to detect adversarial inputs. A reported entropy value above the threshold signifies the network's uncertainty when faced with inputs that it is unfamiliar with or have been otherwise corrupted.

4. Experiments

Our primary experimental objective is to demonstrate the robustness of the KCN in comparison to the CapsNet when faced with both white and black box FGSM attacks with varying $\epsilon \in [0, 1]$. To isolate the effect that the non-parametric GP layer has on the performance of the KCN, we also compare our final architecture (CapsNet foundation + GP classification layer + decoder network) to a version of KCN that forgoes the auxiliary reconstruction network, featuring only the GP classification layer. We refer to this model as KCN-GP.

The model used to develop black box FGSM attacks is a simple two-layer CNN. Specifics about each model architecture and the training/testing protocols are included in the

appendix, Section A.

4.1. Datasets

We evaluate each model described above against MNIST (LeCun et al., 2010), SVHN (Netzer et al., 2011), and CIFAR10 (Krizhevsky & Hinton, 2009). Details about train/test splits, training augmentation, and other preprocessing are included in Section A. All reported results are from performance on the test set.

4.2. Detecting Adversarial Perturbations

Figure 2 shows the nominal $\ell 2$ distances and entropies of a trained KCN model before and after perturbation (dark and light shading, respectively; $\epsilon = 0.3$) for the MNIST test set. The dashed vertical line indicates a notional threshold drawn to allow a 5% false alarm rate. Histograms for other datasets and values of epsilon are included in Section C.

Immediately, one can recognize that the perturbed distribution of $\ell 2$ is separated from the unperturbed values. Note that the threshold drawn for the entropy values is near zero, leaving a long tail of unperturbed inputs that will almost always mix with the values of perturbed inputs These two observations preliminarily suggest that entropy may not be as helpful of an adversarial detection signal compared with the $\ell 2$ reconstruction error. We directly compare the efficacy of each detection method through ROC curve analysis in the next section.
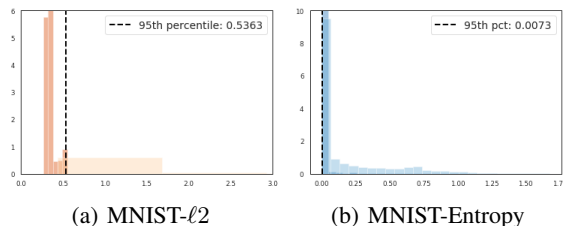

(a) MNIST-$\ell 2$ (b) MNIST-Entropy

Figure 2. MNIST test set detection signals. Darker shading corresponds to unperturbed test examples while the lighter shading corresponds to examples perturbed with FGSM ($\epsilon = 0.3$).

5. Results

The results of our experiments are represented across Figure 3, as well as in Tables 1 and 2 in Section B. For each dataset, the accuracy of the KCN is very close to–and often greater than–that of the CapsNet (refer to Table 1), showing that using a marginal log likelihood as a loss function can effectively train a CapsNet with auxiliary structures (in this case, the GP classifier and decoder network) without sacrificing performance. This indicates that the induced kernel function provides a useful mechanism to flexibly
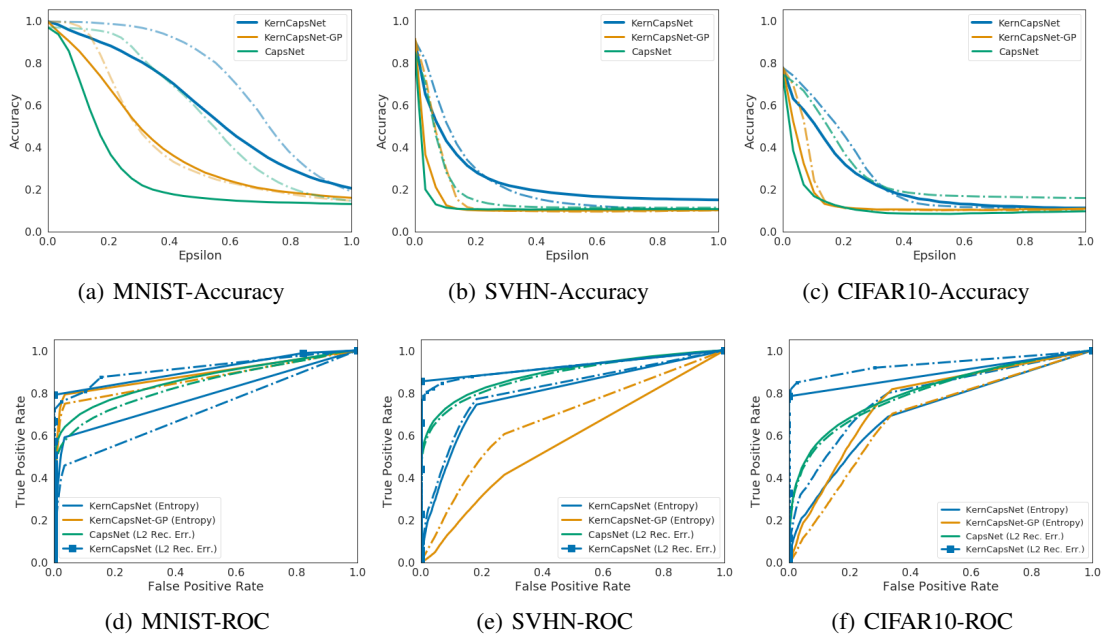
(a) MNIST-Accuracy   (b) SVHN-Accuracy   (c) CIFAR10-Accuracy

(d) MNIST-ROC   (e) SVHN-ROC   (f) CIFAR10-ROC

Figure 3. (a)-(c):Accuracy of the models when facing adversarial perturbations of varying magnitude. (d)-(f):ROC curves for each attack detection signal. (across all images, the dashed line of the same color corresponds to a black box attack on that model)

transform the vector quanitity set of capsule outputs to a Softmax likelihood, rather than relying on hand-tuned loss functions such as the Hinge or Margin losses used in prior work (Sabour et al., 2017; Frosst et al., 2018).

Figure 3(a)-(c) demonstrates that the KCN is more robust than the nominal CapsNet against both white and black box FGSM attacks (solid and dashed lines, respectively) across the three datasets presented in this paper. It appears that the $\ell 2$ reconstruction error serves as an effective regularizer, improving robustness against adversarial perturbations, particularly against black-box attacks (see Figure 4 for example reconstructions made by the KCN).

When evaluating which detection signal is most salient to identifying adversarial perturbations, Figure 3(d)-(f) indicates that the $\ell 2$ reconstruction distance is generally effective distinguishing between perturbed and nominal inputs. AUC metrics for each approach are reported in Table 2. Of note, the $\ell 2$ detection signal from the KCN is stronger than that of the CapsNet, evidenced by the higher reported AUCs.

Although the KCN, by virtue of the GP output layer, does signal that the classifier is uncertain in the presence of adversarially perturbed inputs, entropy alone is not sufficient to determine when an attack occurs due to the overlap of the calculated entropy distributions between normal and perturbed data. The entropy signal could possibly serve as an early warning of model degradation where the $\ell 2$ metric would provide a more dependable signal of whether or not an input to the model has been corrupted.
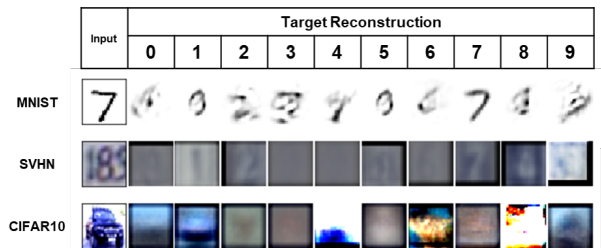

Figure 4. Sample reconstructions from each dataset. MNIST capsules appear to look for "strokes", SVHN and CIFAR10, given limited feature resolution, provide more detail on color and color gradients. Capsules do appear to have "attention" behavior, e.g., the SVHN example, only the 8 is reconstructed.

6. Conclusion

This paper introduces the Kernelized Capsule Network, a model that leverages the flexibility of non-parametric kernel functions to extend the rich representations learned through a CapsNet. We've shown that this approach to training a CapsNet maintains performance, while improving model robustness as well as the ability to detect adversarially perturbed inputs. We are intrigued by the relative simplicity afforded to the training of CapsNets through the GP marginal log likelihood. We believe that hybrid models in the vein of the KCN, or DKL more generally, provide an exciting avenue for future research in model robustness, interpretability and compositionality.

Acknowledgements

References

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., and Goodman, N. D. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 2018.

Bradshaw, J., Matthews, A., and Ghahramani, Z. Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*, 2017.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017.

Frosst, N., Sabour, S., and Hinton, G. DARCCC: Detecting adversaries by reconstruction from class conditional capsules. *arXiv preprint arXiv:1811.06969*, 2018.

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Heckman, J. Sample selection bias as a specification error. *Econometrica: Journal of the econometric society*, pp. 153–161, 1979.

Hensman, J., Matthews, A., and Ghahramani, Z. Scalable variational Gaussian process classification. *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.

Hinton, G., Krizhevsky, A., and Wang, S. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pp. 44–51. Springer, 2011.

Hinton, G., Sabour, S., and Frosst, N. Matrix capsules with em routing. 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Kumar, A. Novel deep learning model for traffic sign detection using capsule networks. *arXiv preprint arXiv:1805.04424*, 2018.

LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2:18, 2010.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Marchisio, A., Nanfa, G., Khalid, F., Hanif, M., Martina, M., and Shafique, M. Capsattacks: Robust and imperceptible adversarial attacks on capsule networks. *arXiv preprint arXiv:1901.09878*, 2019.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Reading digits in natural images with unsupervised feature learning. 2011.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Sabour, S., Frosst, N., and Hinton, G. Dynamic routing between capsules. In *Advances in neural information processing systems*, pp. 3856–3866, 2017.

Sugiyama, M., Krauledat, M., and MÃžller, K.-R. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(May): 985–1005, 2007.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Titsias, M. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pp. 567–574, 2009.

Wang, M., Yu, L., Gan, Q., Zheng, D., Gai, Y., Ye, Z., Li, M., Zhou, J., Huang, Q., Zhao, J., Lin, H., Ma, C., Deng, D., Guo, Q., Zhang, H., Li, J., Smola, A., and Zhang, Z. Deep graph library, 2018. URL http://dgl.ai4.

Wilson, A. and Nickisch, H. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *International Conference on Machine Learning*, pp. 1775–1784, 2015.

Wilson, A., Hu, Z., Salakhutdinov, R., and Xing, E. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pp. 2586–2594, 2016.

A. Training details

Our models are built using PyTorch (Paszke et al., 2017). Deep Graph Library (Wang et al., 2018) is used to solve the dynamic routing problem for Capsule Networks. The probabilistic framework Pyro (Bingham et al., 2018) is used for Deep Kernel Learning. For training we utilize the standard train and test sets for each dataset. We transform the datasets by random shifting images by up to 4 pixels and then apply normalization. The normalizing is calculated using the standard values of mean and standard deviation for each dataset: MNIST has $\mu = 0.1307$ and $\sigma = 0.3081$, CIFAR10 has $\mu = (0.5071, 0.4867, 0.4408)$ and $\sigma = (0.2675, 0.2565, 0.2761)$, and SVHN has $\mu = (0.5, 0.5, 0.5)$ and $\sigma = (0.5, 0.5, 0.5)$. The architecture and training protocol are outlined below.

A.1. Architecture

The capsule network, $\mathcal{I}(x)$, consists of two simple convolutional layers that are unique to each dataset. For MNIST we use C$(1, 12, 4, 2, 3)$ - C$(12, 16, 3, 2, 1)$ where C(n, k, s, p) is a convolution layer followed by a rectified linear unit (ReLU) non-linearity for all but the last layer with parameters of n output channels, kernels of size $k \times k$, stride s, and padding p. The output feature map of these convolutional layers are passed to the Primary Capsule Layer where there are 8 convolutional capsules, $\{C_i(16, 32, 8, 2, 0)\}_{i=1}^{8}$. Each capsule is then stacked and flatten so that there are 8 capsule vectors followed by the squashing non-linearity defined in Sabour et al. (2017). Next we compute the output capsules using the dynamic routing by agreement algorithm implemented in the Deep Graph Library (Wang et al., 2018). This produces the output capsules $v_j \in \mathbb{R}^k$ where $k = 16$ for each class $j = 1 \ldots N_c$ where $N_c = 10$ for each of the datasets. For SVHN and CFIAR10 the convolutional layers are C$(3, 64, 4, 2, 1)$ - C$(64, 64, 3, 2, 1)$ with convolutional capsule layers defined by $\{C_i(64, 32, 8, 2, 0)\}_{i=1}^{8}$. See Figure 1.

The reconstruction layer, see Figure 5 is a set of linear layers of sizes L$(kN_c, 512)$-L$(512, 1024)$-L$(1024, i_c i_h i_w)$ where $i_{c,h,w}$ are the number of image channels and the image height and width respectively. The first two linear layers are followed by the ReLU non-linearity.

The GP classification layer utilizes the RBF kernel (Equation 1) that expects a vector dimension of kN_c, the total number of elements across all capsules. The number of inducing variables is set to 70 and the output size of the variational distribution $q(u)$ is set to the number of class N_c.
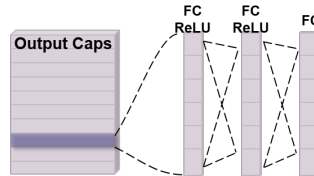

Figure 5. Decoder structure used to reconstruct the input from the Output Capsule layer representation.

A.2. Training Protocol

For each minibatch Pyro (Bingham et al., 2018) provides the ELBO loss, defined by the function "Trace Mean Field ELBO", for deep kernel learning. In addition to the ELBO loss, we add the reconstruction loss using a multiplicative factor of 100 due to the different scales between the losses. The Adam (Kingma & Ba, 2014) optimizer is used with a learning rate of $1e-2$ and default PyTorch parameters.

B. Additional KCN Performance Comparisons

See Table 1 and Table 2 below for performance comparisons.

C. Histograms

We present additional histograms of the $\ell2$ reconstruction error and entropy values that can be used to distinguish perturbed inputs from nominal, unperturbed inputs. In Figure 6 and Figure 7 we present the histograms of the normal, unperturbed inputs and inputs perturbed by FGSM with $\epsilon = 0.3$ for SVHN and CIFAR10 respectively. Figure 8 and Figure 9 show, for MNIST when using a KCN model, how the distribution of $\ell2$ reconstruciton errors and Entropy values change as ϵ increases. From these images, it is easier to gain an intuition into how effective a chosen threshold can be in detecting adversarially perturbed inputs. For $\ell2$ the distributions separate quite early and completely, making it possible for an effective choice in threshold to be made. This task becomes much harder when considering the entropy values as there is significant overlap of the heavy tails of the normal, unperturbed distribution. This complicates any choice of threshold as there will likely be several nominal inputs that will be incorrectly classified as adversarial.

Table 1. **Summary of Training Results**

	MNIST			SVHN			CIFAR10		
	Accuracy	Avg. Ent.	Avg. $\ell2$	Accuracy	Avg. Ent.	Avg. $\ell2$	Accuracy	Avg. Ent.	Avg. $\ell2$
CapsNet	96.7	–	.349	85.0	–	.114	74.3	–	.455
KCN-GP	99.3	.0062	–	91.4	.059	–	77.7	.068	–
KCN	99.4	.0067	.339	90.6	.033	.220	77.6	.061	.977

Table 2. **Summary of Perturbation Detection AUC**

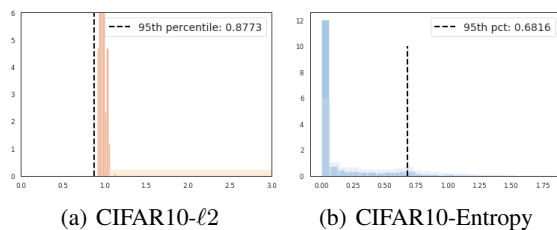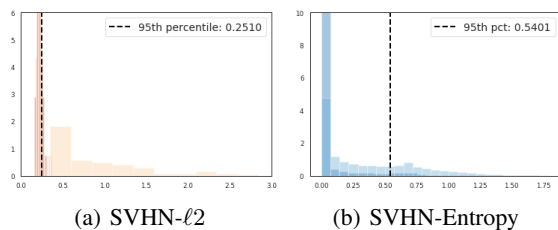|	MNIST		SVHN		CIFAR10	
	White Box	Black Box	White Box	Black Box	White Box	Black Box
CapsNet ($\ell2$)	0.8616	0.8344	0.8907	0.8813	0.8073	0.7992
KCN ($\ell2$)	**0.9072**	**0.9160**	**0.9266**	**0.9229**	**0.8915**	**0.9350**
KCN (Entropy)	0.7806	0.7134	0.7953	0.8132	0.7062	0.7843
KCN-GP (Entropy)	0.8860	0.8631	0.5688	0.6758	0.7580	0.6844

(a) SVHN-$\ell2$ (b) SVHN-Entropy

Figure 6. SVHN test set detection signals. Darker shading corresponds to unperturbed test examples while the lighter shading corresponds to examples perturbed with FGSM ($\epsilon = 0.3$)

(a) CIFAR10-$\ell2$ (b) CIFAR10-Entropy

Figure 7. CIFAR10 test set detection signals. Darker shading corresponds to unperturbed test examples while the lighter shading corresponds to examples perturbed with FGSM ($\epsilon = 0.3$).

Figure 8. MNIST Compilation of $\ell 2$ Reconstruction Error Histograms for all values of ϵ. The two lighter shades correspond to both white and black box attacks.

Figure 9. MNIST Compilation of Entropy Histograms for all values of ϵ. The two lighter shades correspond to both white and black box attacks.