Workshop I: Individual Vehicle Autonomy: Perception and Control
Part of the Long Program
Mathematical Challenges and Opportunities for Autonomous Vehicles

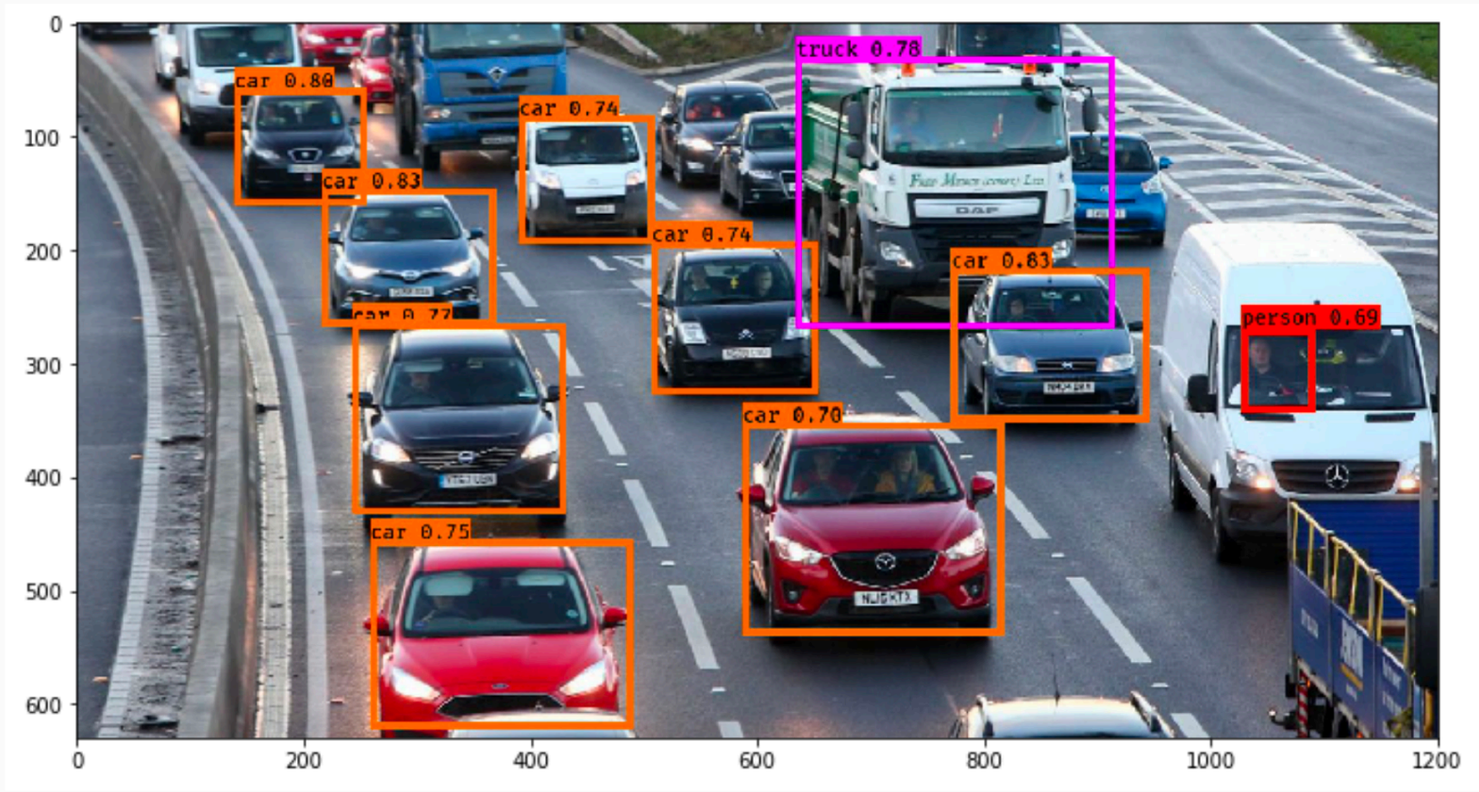
Contributions to deep learning using a mathematical approach: improved model uncertainty, certified robust models, and faster training of Neural ODEs.

Adam Oberman, McGill

Confidence / Uncertainty Estimates for Neural Networks

with Tiago Salvador, postdoc, and Alex Iannantuono, undergrad, now MSc UBC

- When a neural network makes a classification, we want to estimate the uncertainty.
- What is the (estimated) probability correct?
- For deep neural networks, this area is still evolving quickly.



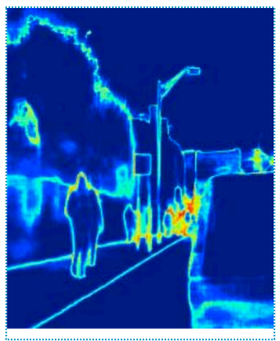
(Example output of a YOLO object detection network, with the probability estimates. Image source: Analytics Vidhya.)

Background story on uncertainty

- There is a lot of confusion about uncertainty for neural network computations.
- Practice: neural networks outputs vector of probabilities coming from softmax. Use these values as class probabilities.
- First glance: this doesn't make sense.
  - Why? For traditional machine learning models, outputs *are* class probabilities.
  - E.g. Spam detection, use conditional probabilities based on appearance of certain words. Bayesian.
  - But this is not the case for neural networks.
- Theory: Dropout in Neural Networks: Yarin Gal, Zoubin Ghahramani. Wiggle the parameters of neural network. Then average the p-values over multiple inferences. Works better when tuned. Bayesian theoretical interpretation(?). I'm confused.

www.reddit.com › MachineLearning › comments › p_h...

[P] How to deal with overconfidence in neural nets ... - Reddit



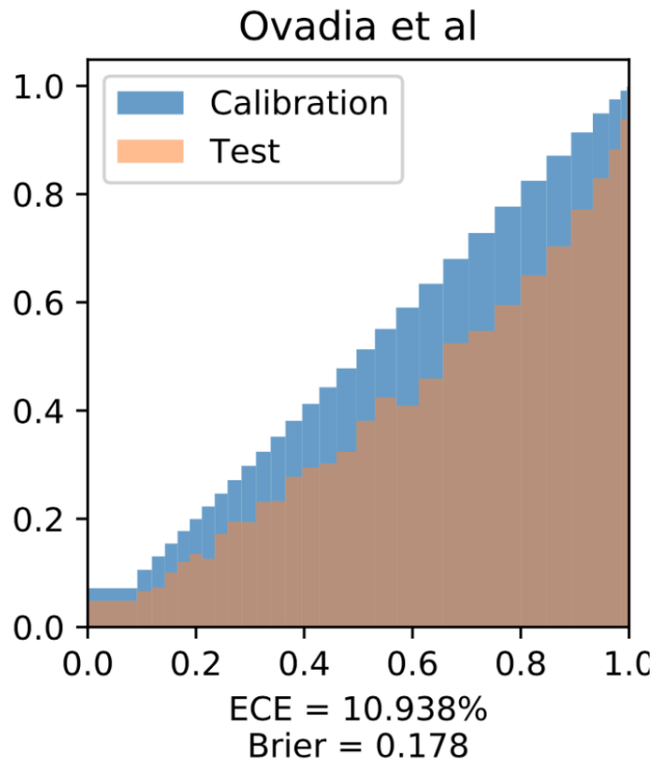
Uncertainty Quantification with Statistical Guarantees in End-to-End Autonomous Driving Control

Deep neural network controllers for autonomous driving have recently benefited from significant performance improvements, and have begun deployment in the real world. Prior to their widespread adoption, safety guarantees are needed on the controller behaviour that properly take account of the uncertainty within the model as well as sensor noise. Bayesian neural networks, which assume a prior over the weights, have been shown capable of

ıral nets? Project. I'm using an ANN built in Keras
come. I have a large ...
/10/20

Background story on uncertainty 2

- Using softmax values as class probabilities seems gives overconfident results, even on test set.
- We proposed to fix this, using simple statistical binning method: converts softmax values into calibrated probabilities.
- I presented this recently at a workshop. Ok.
- But what people really seemed to want was to predict class probabilities on out of distribution (corrupted data) that the network has never seen in training.



Background story on uncertainty 3

- How can we expect performance on unseen corrupted data? Seems impossible.
- E.g. ImageNet-C images. Corrupted with different levels and different types of corruptions.
  - indeed performance degrades severely on these.
- Good contributions (e.g.) AugMix, try to train network with corrupted data.
  - helps, but inevitable trade-offs (worse) on test data.

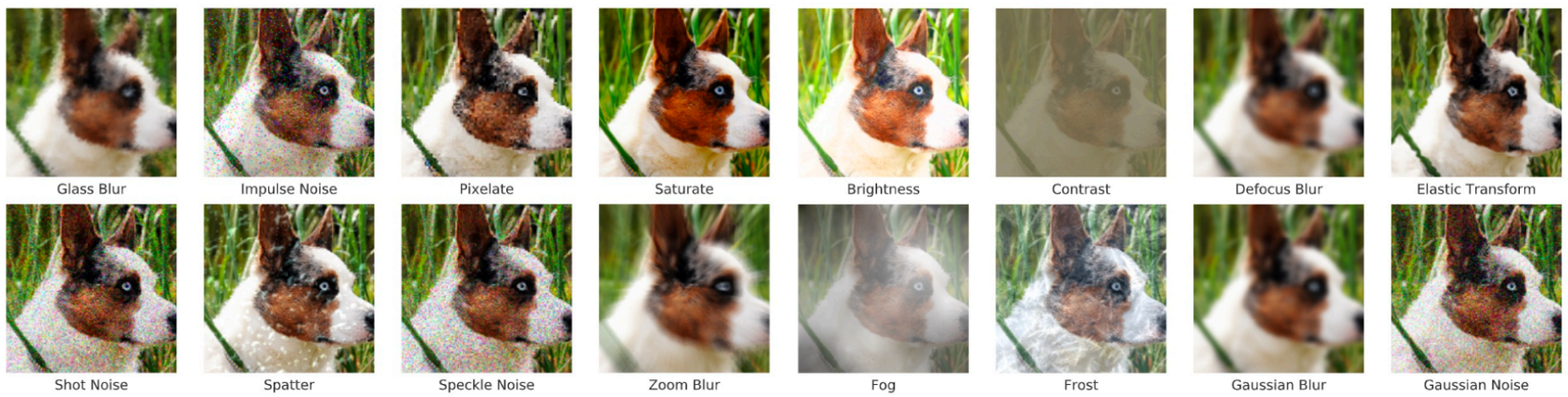
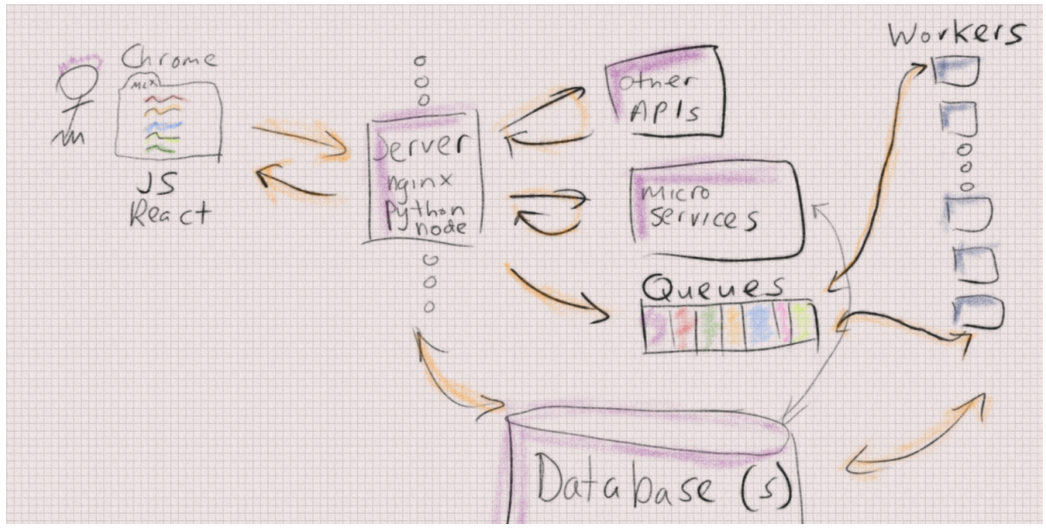


Figure S3: Examples of 16 corruption types in ImageNet-C images, at corruption intensity 3 (on a scale from 1–5). The same corruptions were applied to CIFAR-10. Figure 2 and Section C show boxplots for each uncertainty method and corruption intensity, spanning all corruption types.

Background story on uncertainty 4

- Could imagine a complicated workflow, which
  - detects level of corruption,
  - deploys one of several models, each designed to be tunes to the level of corruption
  - yields most accurate result, based on detection.

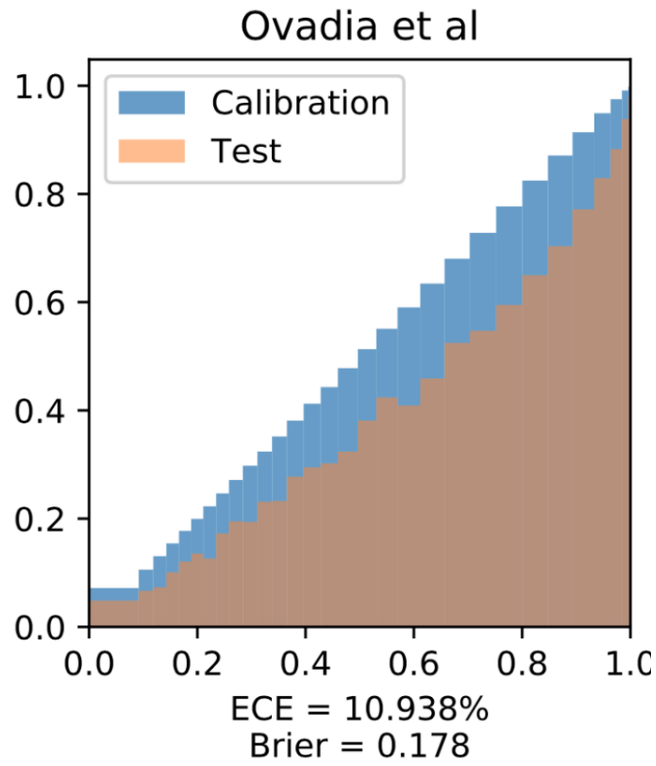


- But people find this too unwieldy.

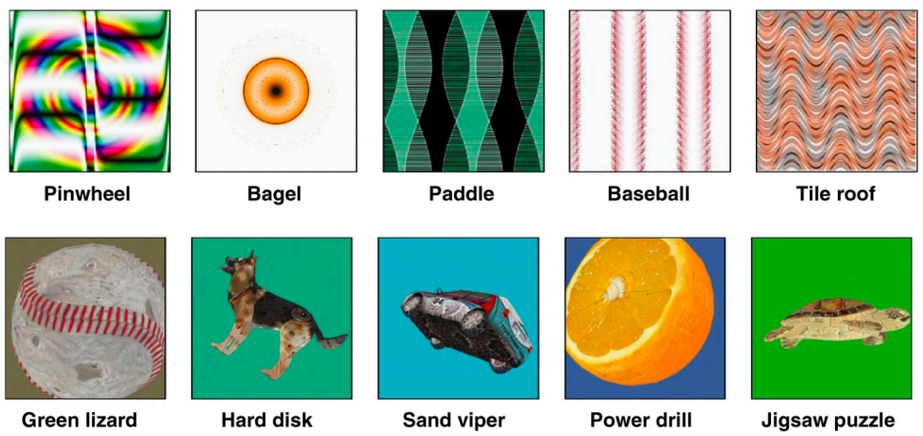
- Our results: Can obtain the benefits of this complex idea for free.
  - Simple enhancement of our basic (in-distribution) calibration method
  - Beat benchmark results on ImageNet-C, across methods and intensity levels
  - Didn't have to retrain any models - just a short lookup table, based on statistical method

Neural Networks are over-confident

- Typically the neural networks are over-confident (even though accuracy is high)
- They are even more over-confident when they are starting to make mistakes/on OOD data
- Calibration error: binning data into predicted confidence, and counting actual errors, shows the overconfidence gap (blue vs. brown).



Confidence (blue) and actual (brown) probability correct for model on shifted distributions.



Overconfidence

- different from Adversarial attacks - images designed to fool neural networks.

Confidence / Uncertainty Estimates for Neural Networks

- Increasingly, models are beings asked to perform on datasets which are different form training datasets: Out of distribution (OOD) data.
- Neural Networks do not perform as well as expected on OOD data

Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study

John R. Zech , Marcus A. Badgeley , Manway Liu, Anthony B. Costa, Joseph J. Titano, Eric Karl Oermann

Published: November 6, 2018 • https://doi.org/10.1371/journal.pmed.1002683

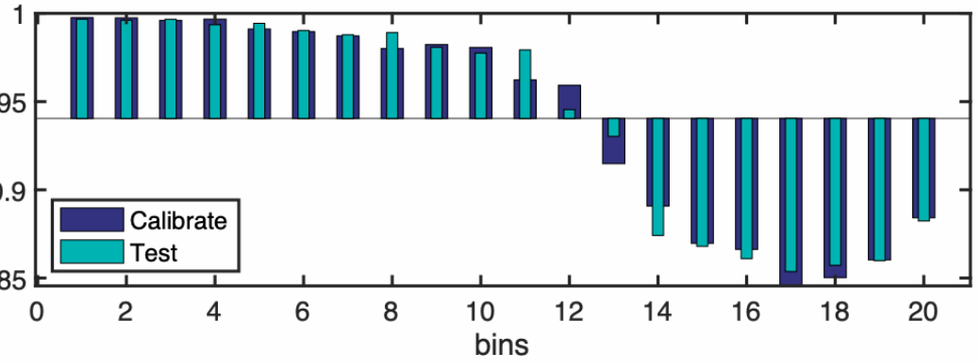
Abstract

Background

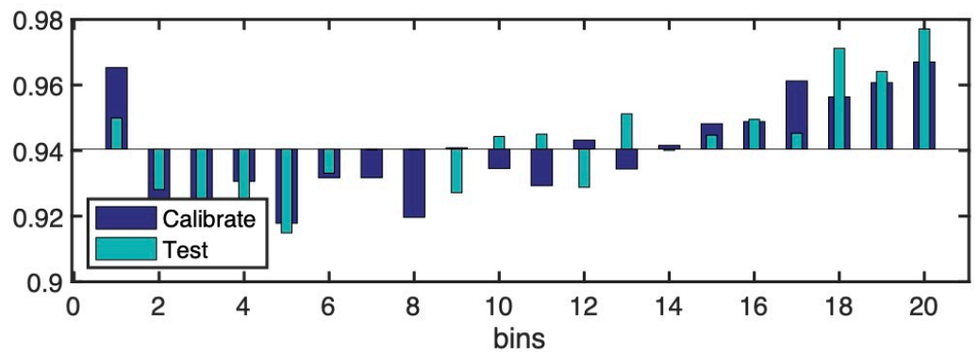
There is interest in using convolutional neural networks (CNNs) to analyze medical imaging to provide computer-aided diagnosis (CAD). Recent work has suggested that image classification CNNs may not generalize to new data as well as previously believed. We assessed how well CNNs generalized across three hospital systems for a simulated pneumonia screening task.

What is Calibration Error?

- Bin images with predicted and actual probability correct. Average error.
- Calibrated: I say 50% chance or rain, and half the time it rains.
- mis-calibrated: I say image is a dog with prob 98%, and it's from the wrong dataset.



well-calibrated: CIFAR model with baseline accuracy .94



poorly-calibrated CIFAR model

- A calibrated model does a good job of predicting uncertainty.
- Note: Model can be accurate, and poorly calibrated, and vice versa.

Benchmarking Methods

- Methods:
  - Basic Idea: use the outputs of network as probabilities.
  - Bayesian Dropout. wiggle the network, average p-values
  - Ensemble methods: take a poll of a few networks, average the p-values
  - (Industry: train an auxiliary network to predict errors)
- Problem: values are not calibrated, and get worse on OOD data.

- (*Vanilla*) Maximum softmax probability (Hendrycks & Gimpel, 2017)
- (*Temp Scaling*) Post-hoc calibration by temperature scaling using a validation set (Guo et al., 2017)
- (*Dropout*) Monte-Carlo Dropout (Gal & Ghahramani, 2016; Srivastava et al., 2015) with rate p
- (*Ensembles*) Ensembles of M networks trained independently on the entire dataset using random initialization (Lakshminarayanan et al., 2017) (we set $M = 10$ in experiments below)
- (*SVI*) Stochastic Variational Bayesian Inference for deep learning (Blundell et al., 2015; Graves, 2011; Louizos & Welling, 2017, 2016; Wen et al., 2018). We refer to Appendix A.6 for details of our SVI implementation.
- (LL) Approx. Bayesian inference for the parameters of the last layer only (Riquelme et al., 2018)

New Benchmark dataset (Ovadia 2019/12)

Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift

Yaniv Ovadia[*] Google Research yovadia@google.com

Emily Fertig[*†] Google Research emilyaf@google.com

Jie Ren[†] Google Research jjren@google.com

Zachary Nado Google Research znado@google.com

D Sculley Google Research dsculley@google.com

Sebastian Nowozin Google Research nowozin@google.com

Joshua V. Dillon Google Research jvdillon@google.com

Balaji Lakshminarayanan[‡] DeepMind balajiln@google.com

Jasper Snoek[‡] Google Research jsnoek@google.com

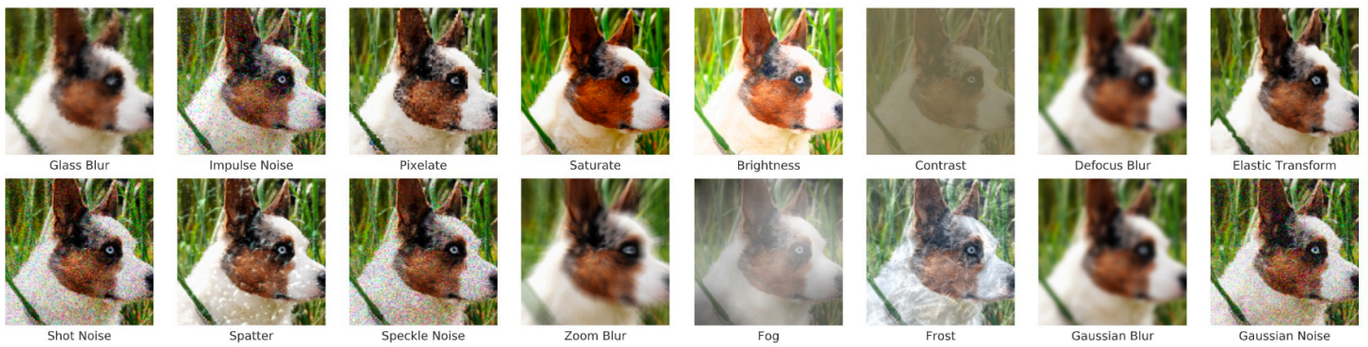
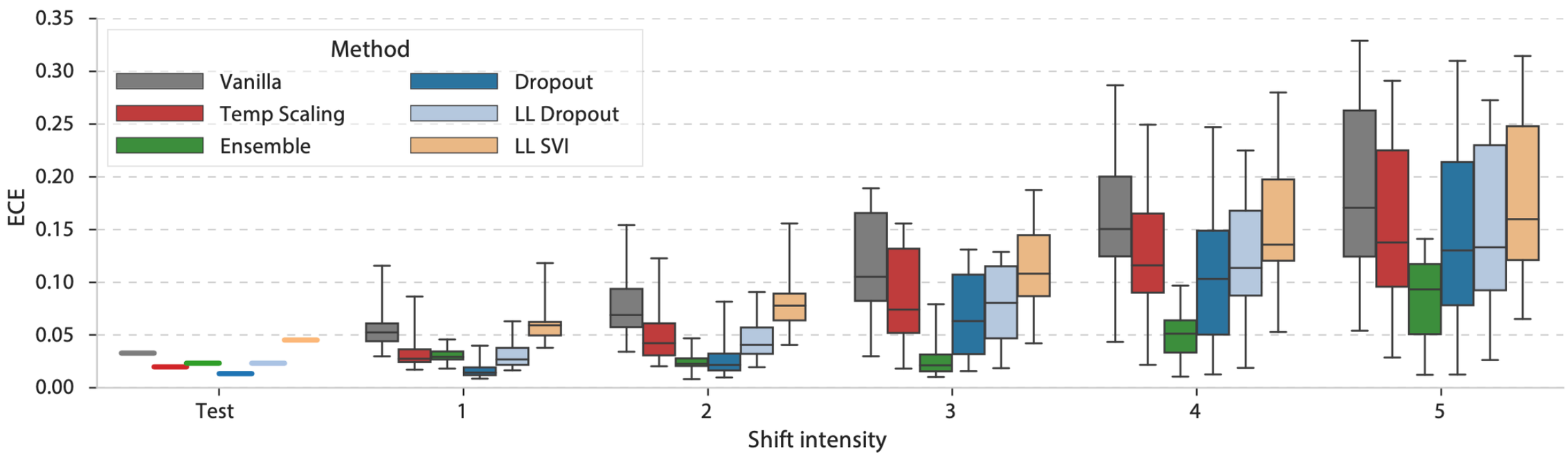
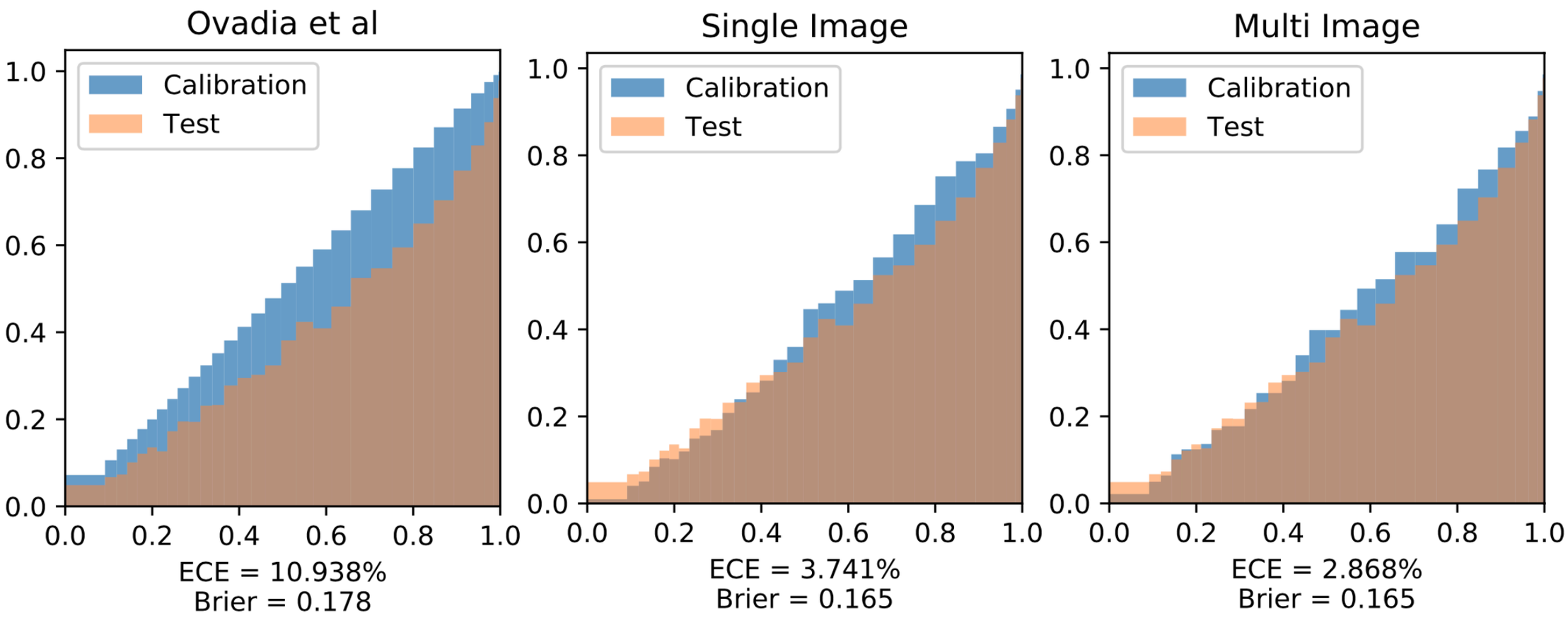


Figure S3: Examples of 16 corruption types in ImageNet-C images, at corruption intensity 3 (on a scale from 1–5). The same corruptions were applied to CIFAR-10. Figure 2 and Section C show boxplots for each uncertainty method and corruption intensity, spanning all corruption types.

- ImageNet with distribution Shift, using 16 types of data corruptions, and 5 different intensities
- Findings: Calibration error gets worse under distributional shift

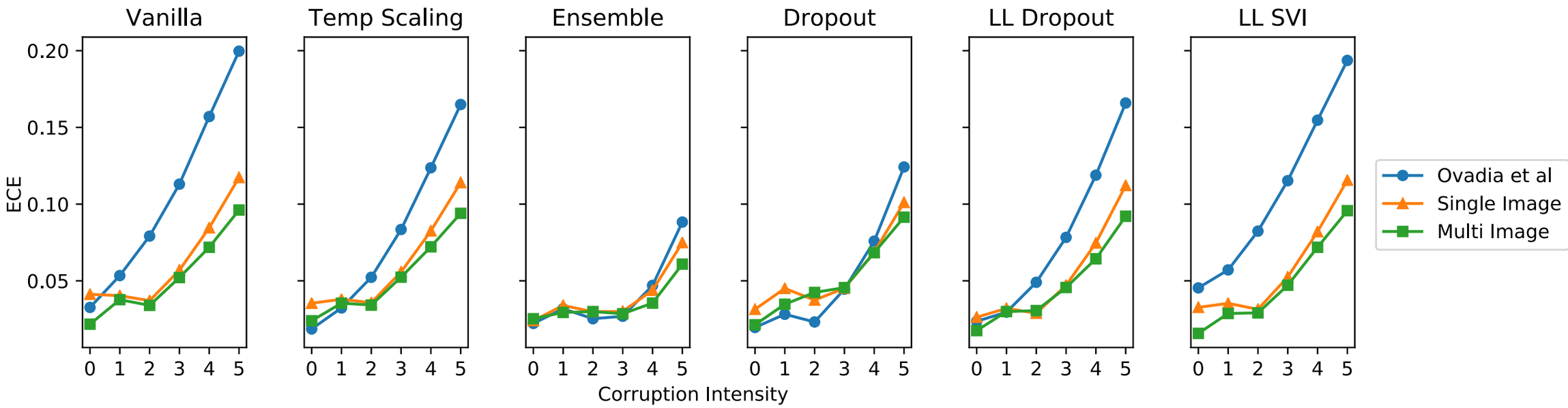


Better calibration



- Calibration error plots: left benchmark method is over confident (ECE = 11%)
- Middle & right, our methods are better calibrated (ECE 3.7 / 2.8 %)
- Our work, led by postdoc Tiago Salvador, beats the benchmark, using simple statistical methods.
- We reduce calibration error, across mild to intense corruption levels

Beating the benchmark



- Figure Mean ECE: benchmark vs. our methods
- The improvement is consistent across methods, with greater improvement at higher corruption levels.
- * Ensemble/Dropout improvement is less, but benchmark dataset provided only summary (average) information, which hindered our calibration methods.

Whisker plot showing means and spread

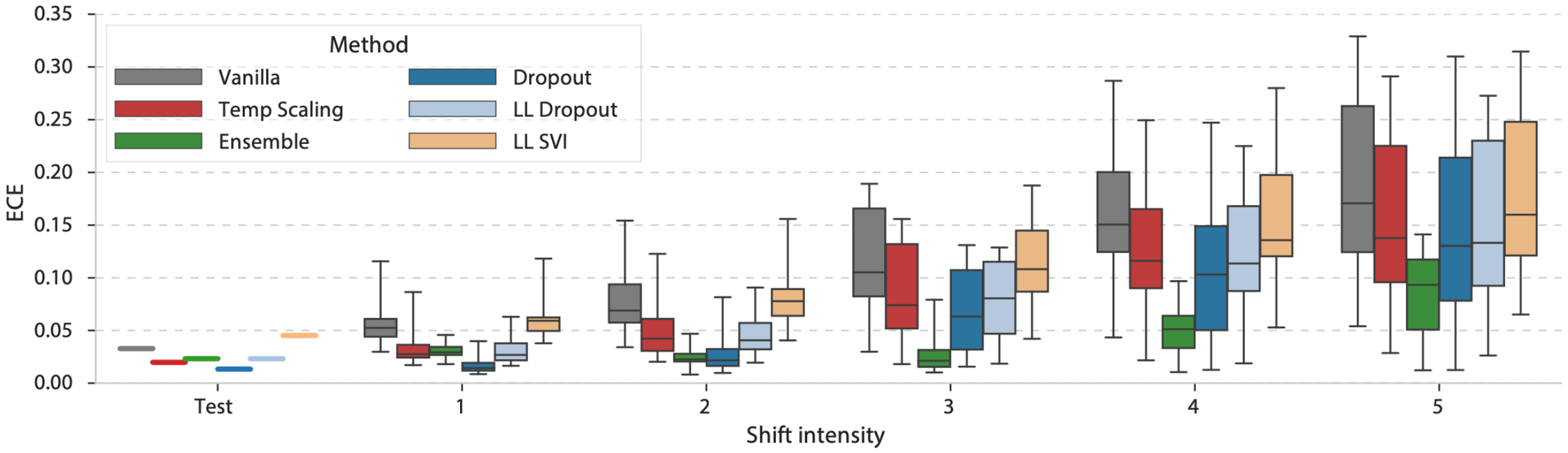
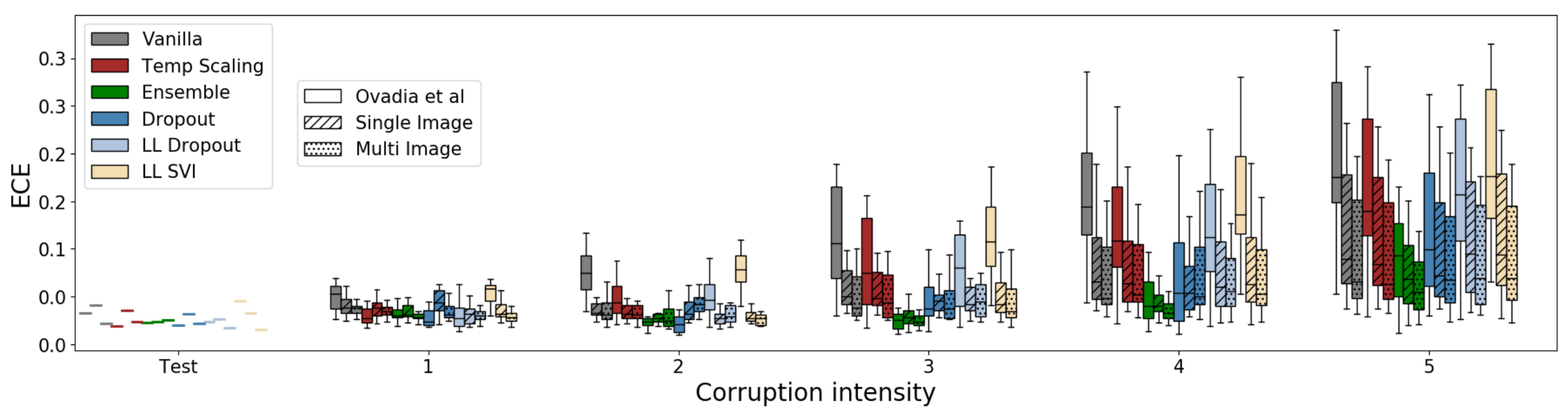


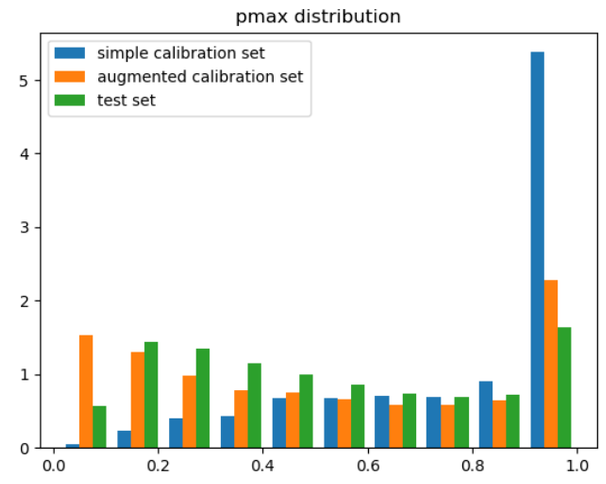
Figure from benchmark paper



- This figure shows the distribution of ECE across different corruptions.
- Compares benchmark with our models.

What's the idea of the method?

- Previous models have been trained a particular dataset (usually clean images).
- But they are calibrated to the dataset they have seen, so they will be overconfident on corrupted images.
- Our idea is to represent *dataset* shift, but *model output* (pmax) shift.
- Expose a model to corrupted images at different intensities, and calibrated each one.
- more corrupted images, have a shift down of pmax values.
- Then, given a single image, estimate the level of corruption (conditional probability) by the pmax value.
- choose the appropriate calibration set, and recalibrate based on that set.
    - with multiple images, can better estimate the level of corruption
- **[Optional: slow down for algorithm details]**

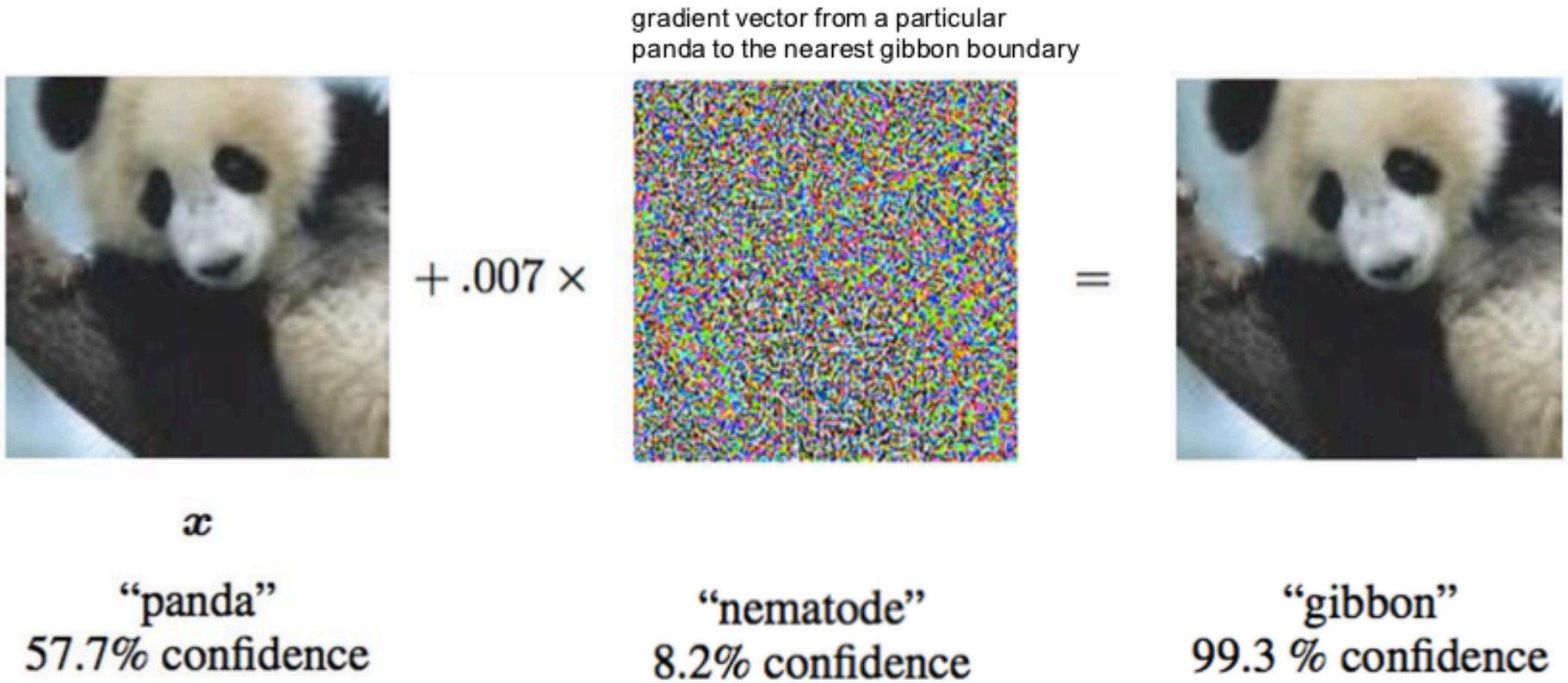


Deterministic Certified Robust Networks

with Chris Finlay PhD, now at Deep Render,
and Ryan Campbell MSc.

- Neural Networks vulnerable to adversarial attacks
- Can train network to be more robust (but lose accuracy)
- Certified networks are guaranteed to be robust to a certain norms: but they have trade-offs:
  - stochastic: require averaging multiple evaluations with noise added.
  - require training new models from scratch
- Our work reduces the trade-off of these certified networks

Intro: Adversarial Attacks



Small (visually imperceptible) perturbations of an image lead to misclassification

Source: EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES, Goodfellow

Attacks on road sign

Left: real graffiti on a stop sign, something that most humans would not think is suspicious. Right: a physical perturbation applied to a stop sign. Models classify the sign on the right as a **Speed Limit: 45 mph sign**!

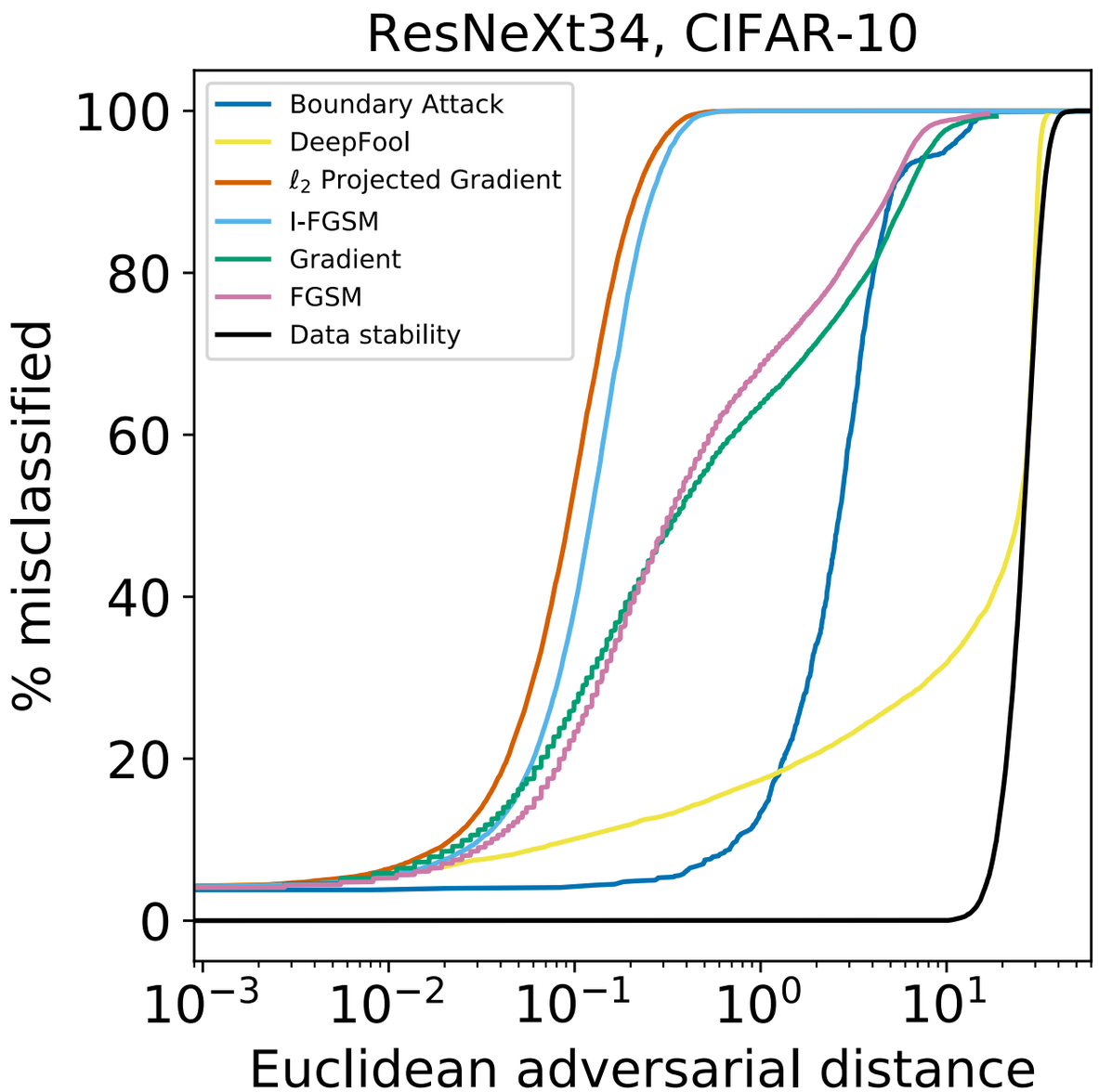
Source: Robust Physical-World Attacks on Deep Learning Visual Classification.

Madry: Defence by adversarial training

- Simple idea: train network replacing original images with attacked images (still using correct labels).
- Now when someone attacks the images, the model has already been trained to recognize them.
- Benefits: improved adversarial robustness
- Problem: loss of accuracy (say from 4% to 12% on CIFAR 10).

Aleksander Mądry

Arms race of attack methods and defences



Error curve: *probability an image misclassified as a function of adversarial attack vector norm.*

Error curve for an undefended model for different attacks.

Practical attacks: attack an image until misclassification occurs. Record distance. What is the difference between the more and less vulnerable images?

Certified Models

Given an input x, the model f is certified to ℓ_2 norm r at x if it gives the same classification on $f(x+\eta)$ for all perturbation η with norm up to r,

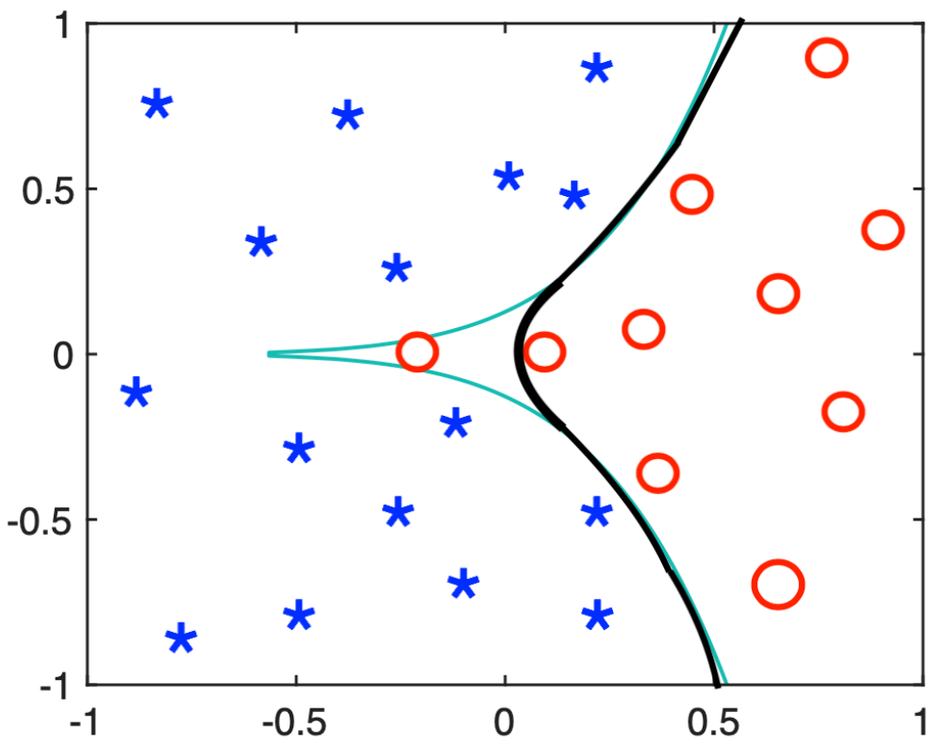
$$\arg\max f(x+\eta) = \arg\max f(x), \qquad \text{for all } \|\eta\|_2 \leq r \tag{1}$$

Cohen et al. (2019) and Salman et al. (2019) certify models by defining a "smoothed" model, f^{smooth}, which is the expected Gaussian average of our initial model f at a given input example x,

$$f^{smooth}(x) \approx \mathbb{E}_\eta \left[f(x+\eta) \right] \tag{2}$$

where the perturbation is sampled from a Gaussian, $\eta \sim \mathcal{N}(0, \sigma^2 I)$. Cohen et al. (2019) used

- Empirically models are not smooth enough
- The model in this form, convolution with a Gaussian, is smoothed.
- But we don't know how to do Gaussian convolution in high dimensions, so need to sample



we want a smoother classification boundary

Challenges with Certification

- Once we know the model is smooth, can certify by sampling the base model (many, many times) in a neighborhood.
- If the majority of the base values agree with classification, can then compute a certified bound for the smoothed model.
- Costs: inference is costly 30X. Certification is costly 1000X.
- Also: lose accuracy when you add Gaussian noise.
- So need to train a model from scratch to be resilient to Gaussian noise. What if we could train a model to be smooth?
- Bottom line: certified models still far from practical.

What if we could train a model to be smooth?

- Essentially, we want to train a model to be the Gaussian convolution of the base model.
- Do this by proposing a loss which is equivalent to Gaussian convolution.

Theorem 1. *(Bishop, 1995) Training a feed-forward neural-network model using the quadratic (or mean-squared error) loss, with added Gaussian noise of mean 0 and variance σ^2 to the inputs, is equivalent to training with*

$$\mathbb{E}_x \left[\|f(x) - y\|^2 + \sigma^2 \|\nabla f(x)\|^2 \right] \tag{4}$$

up to higher order terms.

- Previously models were trained with noise to add a regularization.
- We go the other direction: add regularization to obtain a smoother model.
- use a modified loss for efficiency

$$\mathbb{E}_x \left[\frac{1}{2} \left\| \text{softmax}\left(f^{smooth}(x) \right) - \text{softmax}\left(f(x) \right) \right\|_2^2 + \frac{\sigma^2}{2} \left\| \nabla_x f^{smooth}(x) \right\|_2^2 \right]$$

Gaussian smoothing via PDE regularization

Theorem 2. *(Strauss, 2007) Let f be a bounded function, $x \in \mathbb{R}^d$, and $\eta \sim \mathcal{N}\left(0, \sigma^2 I\right)$. Then the following are equivalent:*

1. $\mathbb{E}_\eta\left[f(x+\eta)\right]$, *the expected value of Gaussian averages of f at x.*
2. $\left(f * \mathcal{N}(0, \sigma^2 I)\right)(x)$, *the convolution of f with the density of the $\mathcal{N}(0, \sigma^2 I)$ distribution evaluated at x.*
3. *The solution of the heat equation,*

$$\frac{\partial}{\partial t} f(x,t) = \frac{\sigma^2}{2} \Delta_x f(x,t) \tag{6}$$

at time $t = 1$, with initial condition $f(x, 0) = f(x)$.

- using PDE fact that Gaussian convolution is equivalent to heat equation.
- use a modified loss for efficiency
- Can start with a pre-trained model, and retrain for 1 epoch (much less)

$$\mathbb{E}_x\left[\frac{1}{2}\left\|\text{softmax}\left(f^{smooth}(x)\right) - \text{softmax}\left(f(x)\right)\right\|_2^2 + \frac{\sigma^2}{2}\left\|\nabla_x f^{smooth}(x)\right\|_2^2\right]$$

Results

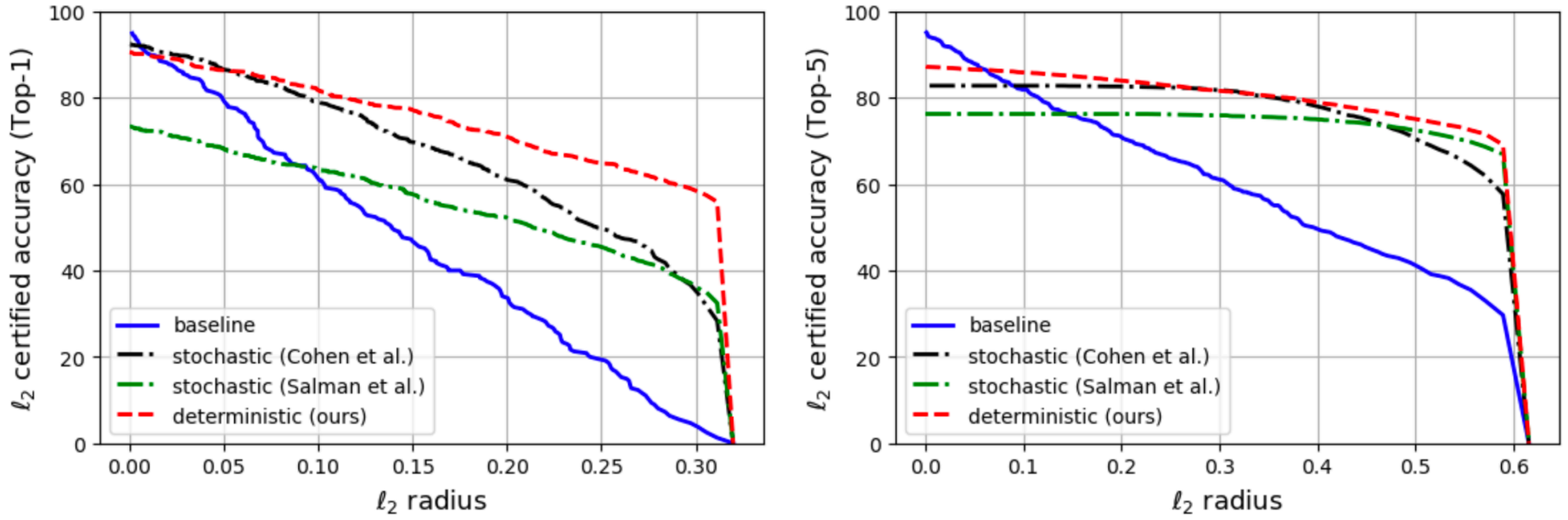
- Faster to train, faster to evaluate.
- Certified

Table 1: *A comparison of robust models. Stochastic smoothing arises from methods like the ones presented in Cohen et al. (2019) and Salman et al. (2019). Adversarial training from Madry et al. (2018).*

Model	Can be obtained from any pretrained model	Evaluation in one forward pass	Is certified?
Deterministic Smoothing (ours)	✔	✔	✔
Stochastic Smoothing	✗	✗	✔
Adversarial Training	✗	✔	✗

Table 2: *Average classification inference time (seconds)*

Model	CIFAR-10		ImageNet-1k	
	CPU	GPU	CPU	GPU
Deterministic (ours)	0.0049	0.0080	0.0615	0.0113
Stochastic (Cohen et al., 2019)	0.0480	0.0399	0.1631	0.0932



(a) *CIFAR-10 top-1 certified accuracy,* $\sigma = 0.10$ **(b)** *ImageNet top-5 certified accuracy,* $\sigma = 0.25$

Figure 1: *Certified accuracy as a function of* ℓ_2 *radius.*

- Better certified bounds that existing models.
- Can smooth an adversarially trained model, and get 4X certified bounds

Table 4: ℓ_2 *certified radii summary statistics for robust models on ImageNet-1k*

Model	ℓ_2 radius		
	median	mean	max.
Certified adversarially trained	0.4226	0.4193	0.6158
Adversarially trained	0.0790	0.1126	0.6158
Undefended baseline	0.0	0.1446	0.6158

Implementation details

- Training models with gradient regularization would normally be very costly.
- We use two ideas to make it faster:
  - Finite differences for the gradient (so no double back-propagation):detach this term from the automatic differentiation computation graph
  - The full model gradient is high dimensional, we approximate it with random projections, using concentration of measure.

Note that the $\left\|\nabla_x f^{smooth}(x)\right\|_2^2$ term in (3) requires the computation of a Jacobian matrix norm. In high dimensions this is computationally expensive. To approximate this term, we make use of the *Johnson-Lindenstrauss lemma* (Johnson and Lindenstrauss, 1984; Vempala, 2005) followed by the finite difference approximation from Finlay and Oberman (2019). We are able to approximate $\left\|\nabla_x f^{smooth}(x)\right\|_2^2$ by taking the average of the product of the Jacobian matrix and Gaussian noise vectors. Jacobian-vector products can be easily computed via reverse mode automatic differentiation, by moving the noise vector w inside:

$$w \cdot (\nabla_x v(x)) = \nabla_x \left(w \cdot v(x)\right) \tag{5}$$

Part 2 :Faster Training of Neural ODEs using OT regularization

How to train your neural ODE: the world of Jacobian and kinetic regularization, ICML 2020

- Chris Finlay, Postdoc McGill, now at DeepRender

- Levon Nurbekyan, Postdoc McGill, now postdoc UCLA w. Stan Osher

- Jorn-Henrik Jacobsen, Vector Institute, University of Toronto, now Senior Research Scientist at Apple.

- Background:
  - Generative Models Results
  - *Neural ODEs*
- Regularization
  - Improvements to Training Time
  - Example Images

Background: Density Estimation and Generative Models (GANs)

- GANs (Goodfellow) 21,000 citations since 2014!
- First method to generate realistic looking new images from large sample of images.

Ian Goodfellow

FOLLOW

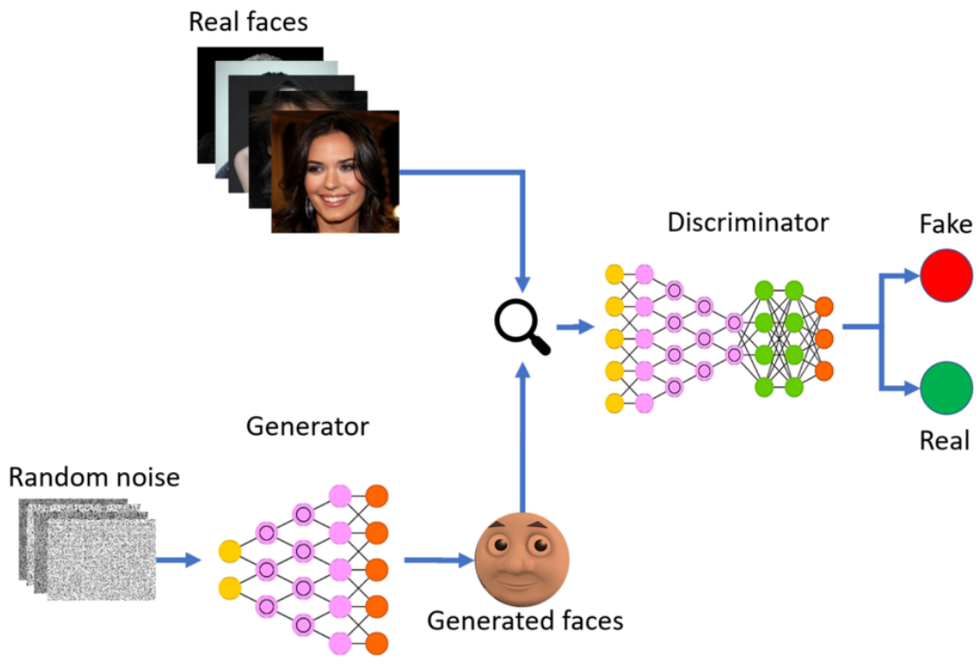
Unknown affiliation

Verified email at cs.stanford.edu - Homepage

Deep Learning

ARTICLES CITED BY CO-AUTHORS

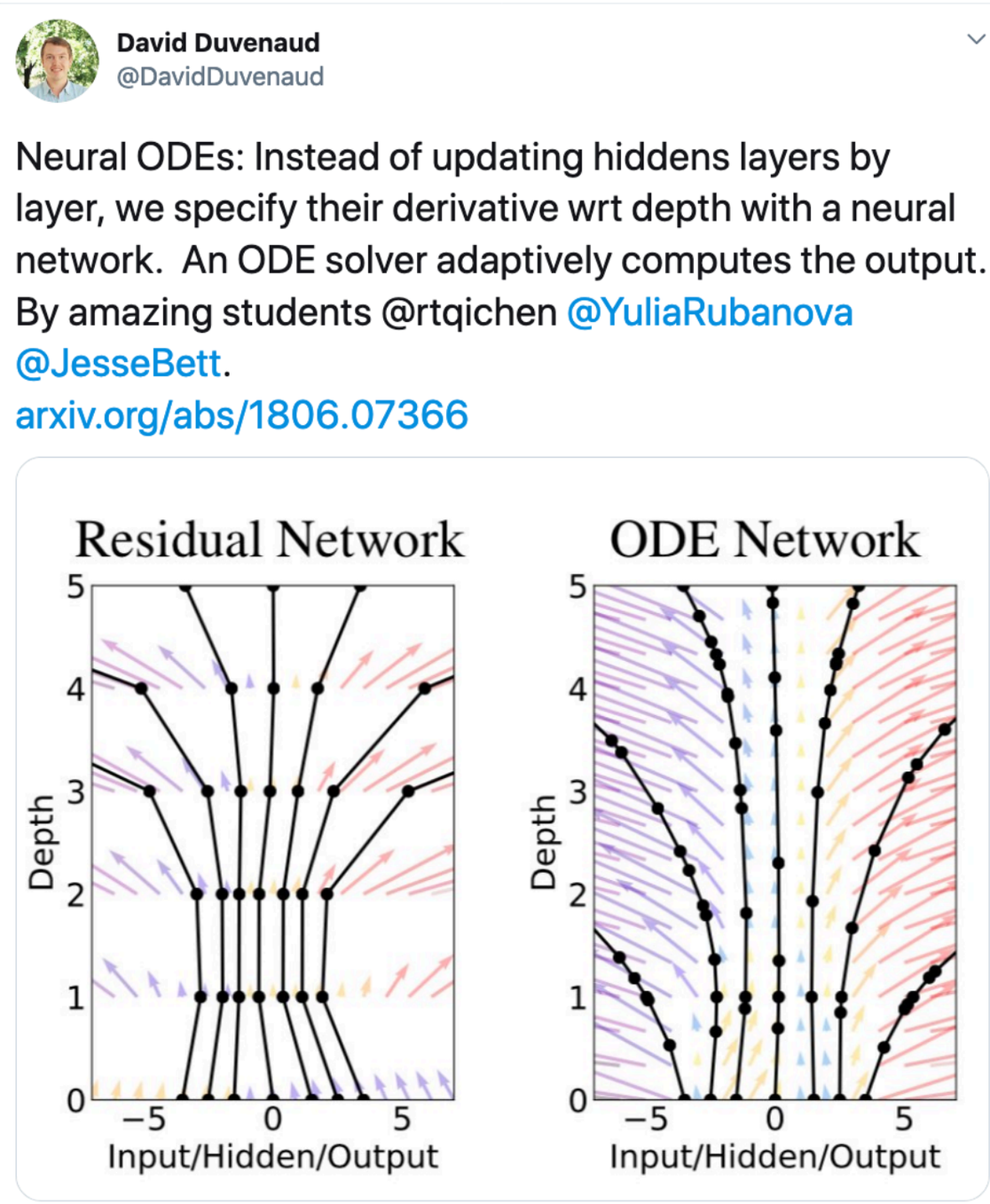
TITLE	CITED BY	YEAR
Generative adversarial nets I Goodfellow, J Pouget-Abadie, M Mirza, B Xu, D Warde-Farley, S Ozair, ... Advances in neural information processing systems, 2672-2680	21440	2014



Images from NVIDIA GAN 2018

Neural ODEs for generative models

- GANs: High quality images, but suffer from "mode collapse", in worst case, can just memorize samples. W-GANs: enforce 1-Lipschitz, but still not onto.
- Alternative: use an invertible network
  - train by mapping images to noise
    - generate by mapping noise to images.
- Normalizing flows: E. Tabak (Courant Math), 2013,
- Neural ODEs (classification), Duvenaud, 2018 (Vector Inst, Toronto) and Haber Ruthotto, 2018 (math).
- FFJORD: (neural ODEs for generative models) Duvenaud, ICLR 2019. Generative Neural ODEs



Neural ODEs for Generative Models

- Math Involved:
  - ODE architecture (adaptive ResNet)
    - Invertible network
    - Divergence computation in high dimensions
- Appealing architecture: more mathematical, more abstract.

Density estimation

To learn the distribution p_θ, solve the following log-likelihood optimization problem

$$\max_\theta \sum_i \log p_{\mathcal{N}}(z_i(T)) + \int_0^T \operatorname{div}(v_\theta)(z_i(s), s)\,\mathrm{d}s$$

where $z_i(s)$ satisfies the ODE

$$\begin{cases} \dot{z}_i(s) = v_\theta(z_i(s), s) \\ z_i(0) = x_i \end{cases}$$

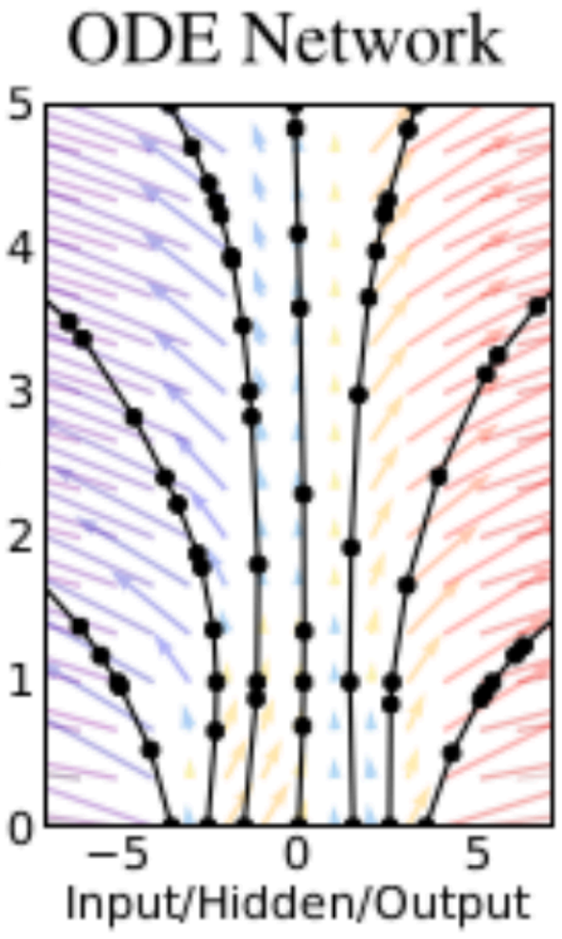
Generation

Sample $z \sim \mathcal{N}$, and solve

$$\begin{cases} \dot{x}(s) = -v_\theta(x(s), s) \\ x(0) = z \end{cases}$$

Ie, $x = z + \int_T^0 v_\theta(z(s), s)\,\mathrm{d}s$

Good idea: ODEs for architecture.
Missing idea: penalize large Lipschitz constants.

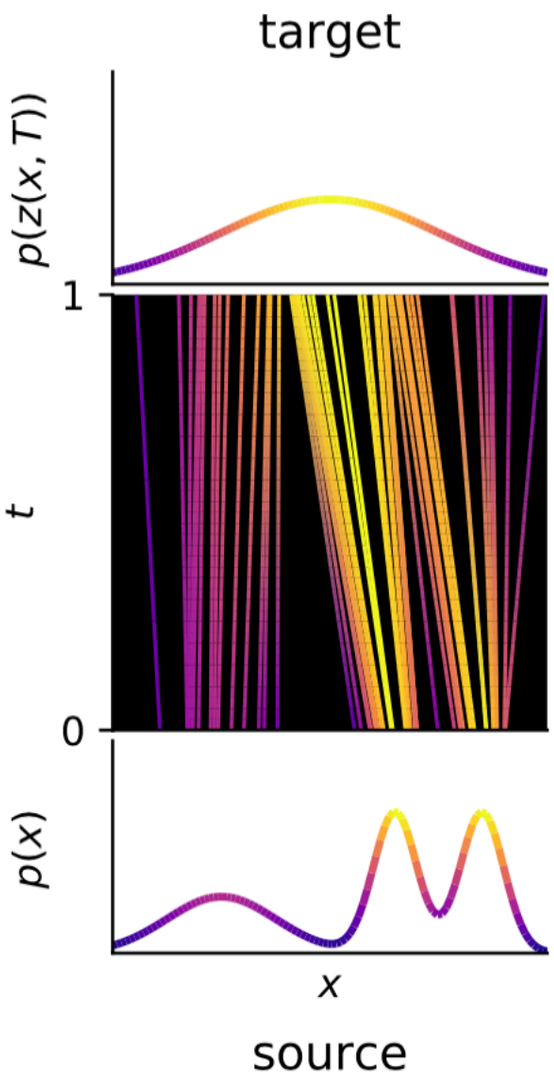


FFJORD is promising, but there are no constraints placed on the paths the particles take

- As long as source (data) distribution is mapped to target (normal) distribution, the log-likelihood is maximized
- ie solutions are not unique
- If the particle paths are "wobbly" the adaptive ODE solver has to take many tiny steps, with many function evaluations. This is time consuming

Notes: no control on paths means no control on Lipschitz constant.
Result: more steps needed for ODE, slower network, and longer training times.

Add regularization to obtain the OT map



Add two terms to the objective to encourage regularity of trajectories:

- $\int_0^T \|v_\theta(x(s), s)\|^2 \, \mathrm{d}s$, the kinetic energy. This is closely related to the Optimal Transport cost between distributions (Benamou-Brenier)
- $\int_0^T \|\nabla_x v_\theta(x(s), s)\|_F^2 \, \mathrm{d}s$, a Frobenius norm penalty on the Jacobian
- Frobenius norms can be computed again with trace estimate:

$$\begin{aligned} \|A\|_F^2 = \mathrm{Tr}(A^T A) &= \mathbb{E}_\eta \left[\eta^T A^T A \eta\right] \\ &= \mathbb{E}\left[\|A\eta\|^2\right] \end{aligned}$$

Connection with Optimal Transportation

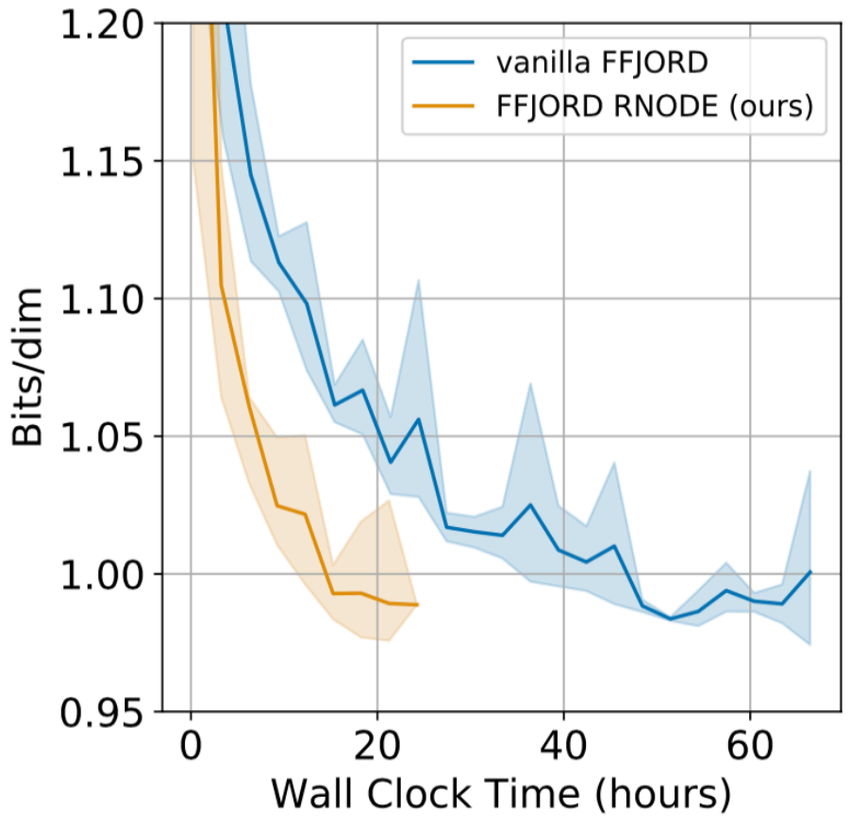
- Benamou-Brenier formulation of Optimal Transportation in Lagrangian coordinates

$$
\begin{aligned}
\min_{\mathbf{f}} \quad & \int_0^T \int \|\mathbf{f}(\mathbf{z}(\mathbf{x},t),t)\|^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x}\mathrm{d}t && \text{(19a)}\\
\text{subject to} \quad & \dot{\mathbf{z}}(\mathbf{x},t) = \mathbf{f}(\mathbf{z}(\mathbf{x},t),t), && \text{(19b)}\\
& \mathbf{z}(\mathbf{x},0) = \mathbf{x}, && \text{(19c)}\\
& \mathbf{z}(\cdot,T)\sharp p = q. && \text{(19d)}
\end{aligned}
$$

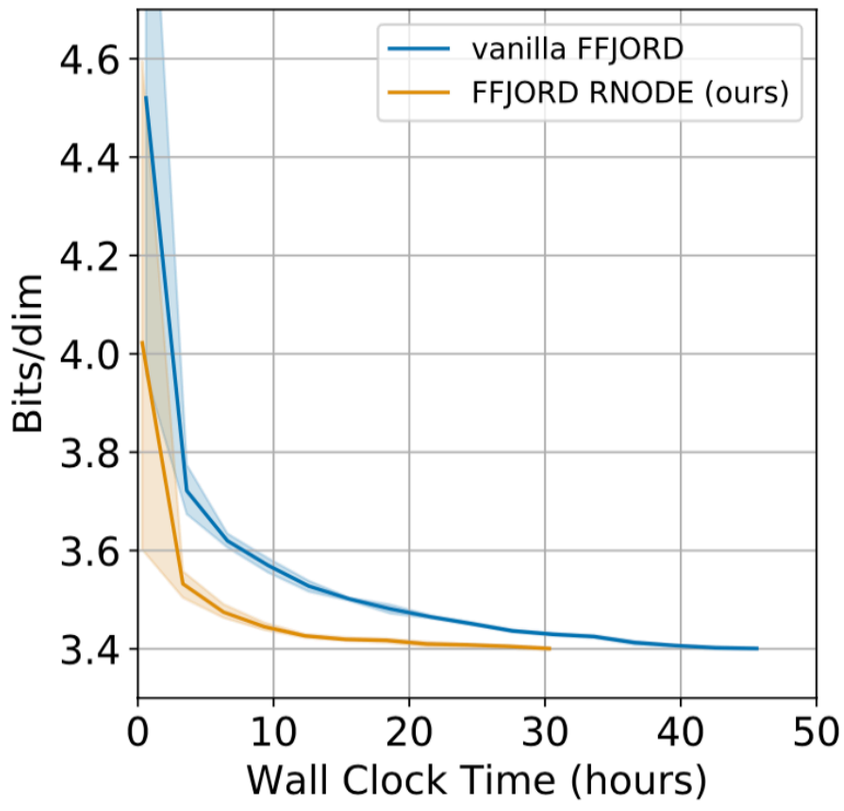
- Our loss, with empirical measures, and a weak form of the push-forward condition

$$
\frac{\lambda}{N}\sum_{i=1}^{N}\int_0^T \|\mathbf{f}(\mathbf{z}(\mathbf{x}_i,t),t)\|^2\,\mathrm{d}t - \frac{1}{N}\sum_{i=1}^{N}\log p_\theta(\mathbf{x}_i)
$$

What is the effect? Faster Training



MNIST



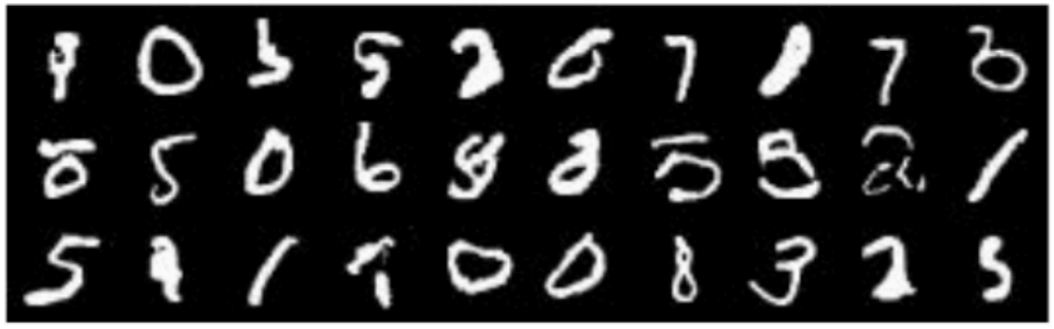
CIFAR10

- Image classification networks which have lots of (heuristic) regularization built in, so the effects of regularization are marginal.
- For neural ODEs, bottleneck was training time. In this case regularization solved the bottleneck. First in class to train on ImageNet.

Generated Images

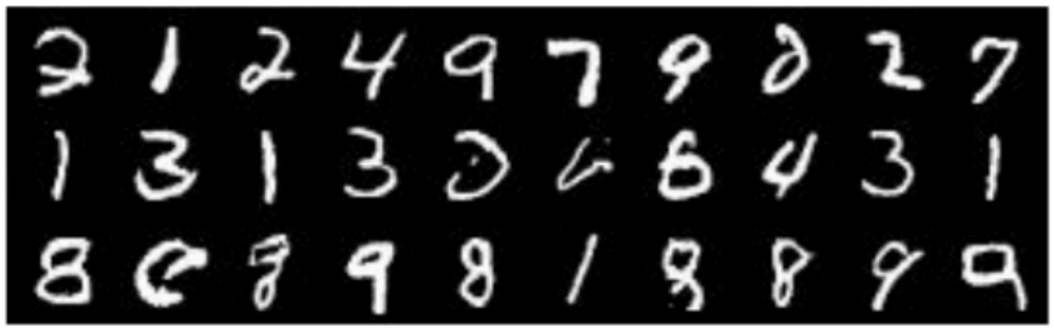
(a) real MNIST images

(b) real CIFAR10 images



(c) vanilla FFJORD

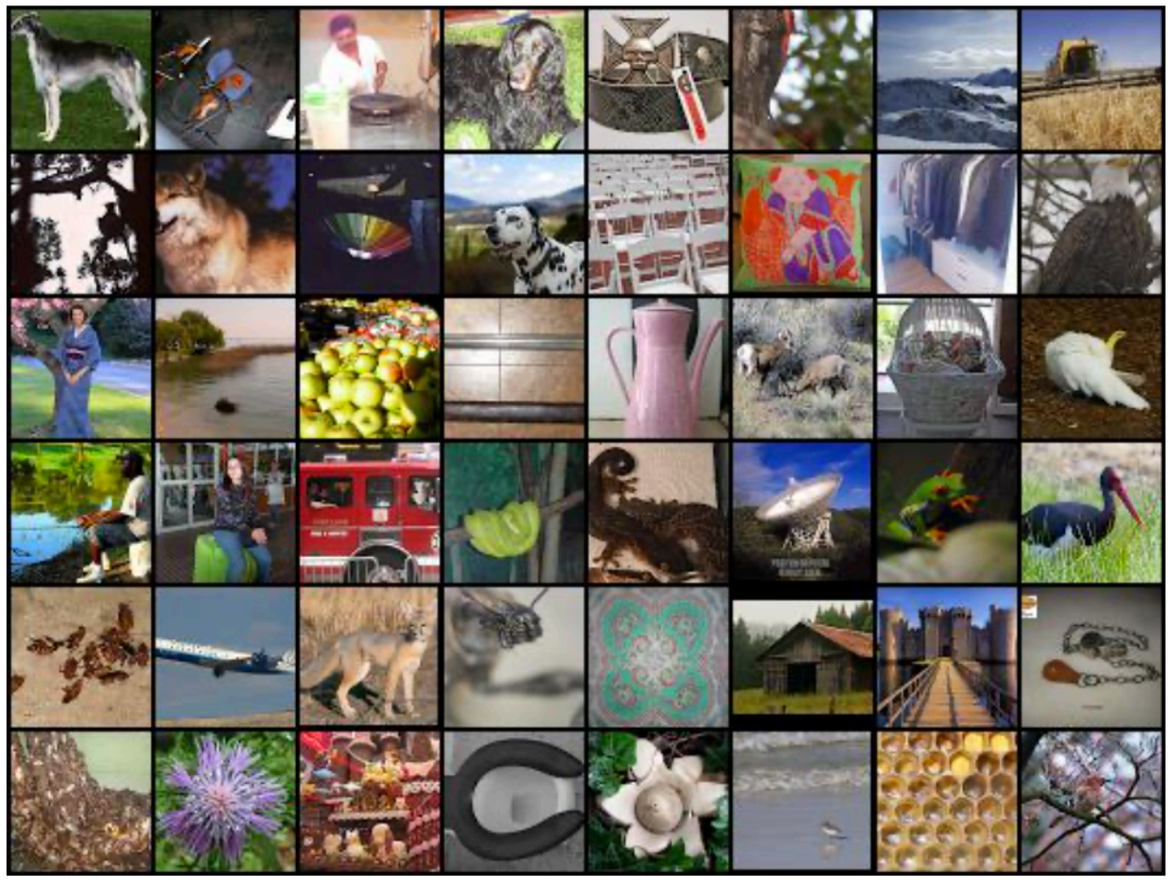
(d) vanilla FFJORD



(e) FFJORD RNODE

(f) FFJORD RNODE

Images



Real ImageNet64 images

Generated ImageNet64 images
(FFJORD RNODE)

Thanks!