Uncertainty & Out-of-Distribution Robustness in Deep Learning

Balaji Lakshminarayanan

balajiln@

Joint work with Akihiro Matsukawa, Alexander Pritzel, Charles Blundell, Dilan Gorur, Eric Nalisnick, Yee Whye Teh

DeepMind

Quantifying Uncertainty In Deep Learning

- Why predictive uncertainty?
  - Good uncertainty scores quantify when we can trust the model’s predictions

Quantifying Uncertainty In Deep Learning

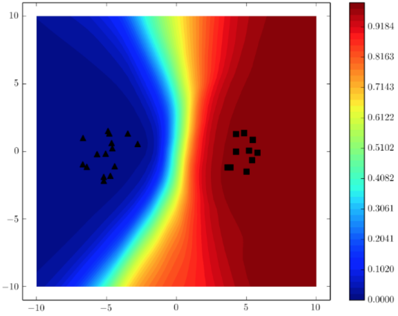
- Why predictive uncertainty?
  - Good uncertainty scores quantify when we can trust the model's predictions
- Predict output distribution $p(y|x)$ rather than point estimate
  - Classification: output label y^* along with confidence
  - Regression: output mean and variance

Source of uncertainty: Inherent stochasticity

Output y for a given x could be inherently stochastic

- Rewards in a casino
- Measurement noise in *y*
- Noise in labeling process (outcome could depend on rater)
- Also known as **aleatoric uncertainty**
- Considered to be “irreducible uncertainty”: persists even in the limit of infinite data

Source of uncertainty: Model uncertainty



- Multiple values of parameters could be consistent with the observed data
- Also known as **epistemic uncertainty**
- Considered to be "reducible uncertainty": vanishes in the limit of infinite data (subject to identifiability)

Applications of Predictive Uncertainty[1]

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)

[1]Weight uncertainty is also useful, e.g. compression, sensitivity analysis

Applications of Predictive Uncertainty[1]

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Active learning for efficient data collection

[1] Weight uncertainty is also useful, e.g. compression, sensitivity analysis

Applications of Predictive Uncertainty[1]

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Active learning for efficient data collection
- Dealing with noisy data and train-test skew in production systems

[1]Weight uncertainty is also useful, e.g. compression, sensitivity analysis

Applications of Predictive Uncertainty[1]

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Active learning for efficient data collection
- Dealing with noisy data and train-test skew in production systems
- Reinforcement learning: (Safe) Exploration

[1] Weight uncertainty is also useful, e.g. compression, sensitivity analysis

Applications of Predictive Uncertainty[1]

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Active learning for efficient data collection
- Dealing with noisy data and train-test skew in production systems
- Reinforcement learning: (Safe) Exploration
- Model interpretability and visualization

[1] Weight uncertainty is also useful, e.g. compression, sensitivity analysis

Applications of Predictive Uncertainty[1]

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Active learning for efficient data collection
- Dealing with noisy data and train-test skew in production systems
- Reinforcement learning: (Safe) Exploration
- Model interpretability and visualization
- Build modular systems that **know what they don't know**

[1] Weight uncertainty is also useful, e.g. compression, sensitivity analysis

Applications of Predictive Uncertainty[1]

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Active learning for efficient data collection
- Dealing with noisy data and train-test skew in production systems
- Reinforcement learning: (Safe) Exploration
- Model interpretability and visualization
- Build modular systems that **know what they don't know**
- ... and many more!

[1]Weight uncertainty is also useful, e.g. compression, sensitivity analysis

How do we measure the quality of uncertainty?

- **Calibration**: Measures how probabilistic forecasts align with observed long-run frequencies
  - Weather forecasting: Of all days where model predicted rain with 80% probability, what fraction did we observe rain?
  - Measures: Calibration curve / Reliability diagrams, Expected calibration error (ECE)

How do we measure the quality of uncertainty?

- **Calibration**: Measures how probabilistic forecasts align with observed long-run frequencies
  - Weather forecasting: Of all days where model predicted rain with 80% probability, what fraction did we observe rain?
  - Measures: Calibration curve / Reliability diagrams, Expected calibration error (ECE)
- **Robustness to dataset shift**: does the system exhibit higher uncertainty on inputs far away from training data?
  - $p(y|x)$ is typically accurate when $x \sim p_{train}(x)$, but can make overconfident errors when asked to predict on **out-of-distribution (OOD)** inputs

How do we measure the quality of uncertainty?

- **Calibration**: Measures how probabilistic forecasts align with observed long-run frequencies
  - Weather forecasting: Of all days where model predicted rain with 80% probability, what fraction did we observe rain?
  - Measures: Calibration curve / Reliability diagrams, Expected calibration error (ECE)
- **Robustness to dataset shift**: does the system exhibit higher uncertainty on inputs far away from training data?
  - $p(y|x)$ is typically accurate when $x \sim p_{train}(x)$, but can make overconfident errors when asked to predict on **out-of-distribution (OOD)** inputs
  - Cross-validation can inflate performance. Need to measure ability of model to reject OOD inputs (e.g. confidence versus accuracy curves).

How do we measure the quality of uncertainty?

- **Calibration**: Measures how probabilistic forecasts align with observed long-run frequencies
  - Weather forecasting: Of all days where model predicted rain with 80% probability, what fraction did we observe rain?
  - Measures: Calibration curve / Reliability diagrams, Expected calibration error (ECE)
- **Robustness to dataset shift**: does the system exhibit higher uncertainty on inputs far away from training data?
  - $p(y|x)$ is typically accurate when $x \sim p_{train}(x)$, but can make overconfident errors when asked to predict on **out-of-distribution (OOD)** inputs
  - Cross-validation can inflate performance. Need to measure ability of model to reject OOD inputs (e.g. confidence versus accuracy curves).
- **Challenges**
  - Lack of ground truth: no "right answer" in some cases
  - Cost of decisions may be difficult to model

How do deep networks fare?

Deep networks are poorly calibrated

On Calibration of Modern Neural Networks

Chuan Guo [*1] Geoff Pleiss [*1] Yu Sun [*1] Kilian Q. Weinberger [1]

Abstract

Confidence calibration – the problem of predicting probability estimates representative of the true correctness likelihood – is important for classification models in many applications. We discover that modern neural networks, unlike those from a decade ago, are poorly calibrated. Through extensive experiments, we observe that depth, width, weight decay, and Batch Normalization are important factors influencing calibration. We evaluate the performance of various post-processing calibration methods on state-of-the-art architectures with image and document classification datasets. Our analysis and experiments not only offer insights into neural network learning, but also provide a simple and straightforward recipe for practical settings: on most datasets, *temperature scaling* – a single-parameter variant of Platt Scaling – is surprisingly effective at calibrating predictions.

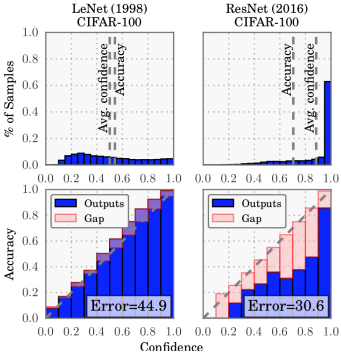


Figure 1. Confidence histograms (top) and reliability diagrams (bottom) for a 5-layer LeNet (left) and a 110-layer ResNet (right) on CIFAR-100. Refer to the text below for detailed illustration.

1. Introduction

High confidence predictions on OOD inputs

Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images

Anh Nguyen
University of Wyoming
anguyen8@uwyo.edu

Jason Yosinski
Cornell University
yosinski@cs.cornell.edu

Jeff Clune
University of Wyoming
jeffclune@uwyo.edu

Abstract

Deep neural networks (DNNs) have recently been achieving state-of-the-art performance on a variety of pattern-recognition tasks, most notably visual classification problems. Given that DNNs are now able to classify objects in images with near-human-level performance, questions naturally arise as to what differences remain between computer and human vision. A recent study [30] revealed that changing an image (e.g. of a lion) in a way imperceptible to humans can cause a DNN to label the image as something else entirely (e.g. mislabeling a lion a library). Here we show a related result: it is easy to produce images that are completely unrecognizable to humans, but that state-of-the-art DNNs believe to be recognizable objects with 99.99% confidence (e.g. labeling with certainty that white noise static is a lion). Specifically, we take convolutional neural networks trained to perform well on either the ImageNet or MNIST datasets and then find images with evolutionary algorithms or gradient ascent that DNNs label with high confidence as belonging to each dataset class. It is possible to produce images totally unrecognizable to human eyes that DNNs believe with near certainty are familiar objects, which we call "fooling images" (more generally, fooling examples). Our results shed light on interesting differences between human vision and current DNNs, and raise questions about the generality of DNN computer vision.

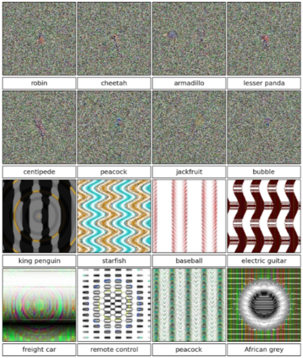


Figure 1. Evolved images that are unrecognizable to humans, but that state-of-the-art DNNs trained on ImageNet believe with $\geq$ 99.6% certainty to be a familiar object. This result highlights differences between how DNNs and humans recognize objects. Images are either directly (*top*) or indirectly (*bottom*) encoded.

A quick overview of Bayesian deep learning

Bayesian deep learning

- Bayesian Model Averaging (BMA) in a nutshell:
  - Specify prior over parameters $p(\theta)$
  - Compute posterior distribution of parameters $p(\theta|\mathcal{D})$
  - Translate parameter uncertainty to predictive uncertainty

Bayesian deep learning

- Bayesian Model Averaging (BMA) in a nutshell:
  - Specify prior over parameters $p(\theta)$
  - Compute posterior distribution of parameters $p(\theta|\mathcal{D})$
  - Translate parameter uncertainty to predictive uncertainty
- True posterior distribution $p(\theta|\mathcal{D})$ is usually hard to compute and approximated:
  - **Variational inference**
    - Approximate true posterior distribution by simpler parametric distribution $q(\theta)$, usually a single mode
    - Laplace approximation, BBB, Dropout variational inference

Bayesian deep learning

- Bayesian Model Averaging (BMA) in a nutshell:
  - Specify prior over parameters $p(\theta)$
  - Compute posterior distribution of parameters $p(\theta|\mathcal{D})$
  - Translate parameter uncertainty to predictive uncertainty
- True posterior distribution $p(\theta|\mathcal{D})$ is usually hard to compute and approximated:
  - **Variational inference**
    - Approximate true posterior distribution by simpler parametric distribution $q(\theta)$, usually a single mode
    - Laplace approximation, BBB, Dropout variational inference
  - **Markov Chain Monte Carlo**
    - Do not make parametric assumption
    - Define Markov chain that eventually samples from true posterior
    - Lots of recent work on stochastic gradient MCMC: SGLD, SGHMC, etc

Why Bayesian deep learning?

- BMA satisfies the axioms of probability (as it uses Bayes rule) and is the only way to protect against “Dutch books”.

Why Bayesian deep learning?

- BMA satisfies the axioms of probability (as it uses Bayes rule) and is the only way to protect against “Dutch books”.
- **BMA is optimal if**:
  - “prior is correct” i.e. true model is within hypothesis class
  - true posterior can be computed exactly

Why Bayesian deep learning?

- BMA satisfies the axioms of probability (as it uses Bayes rule) and is the only way to protect against "Dutch books".
- **BMA is optimal if**:
  - "prior is correct" i.e. true model is within hypothesis class
  - true posterior can be computed exactly
- Bayesian deep learning is gaining popularity
  - Lots of great tools exist for low dimensional problems (e.g. Hamiltonian Monte Carlo, Gaussian processes)
  - Better software and probabilistic programming tools
  - But the true multi-modal posterior is really hard to approximate for high dimensional spaces

Is there a scalable alternative to Bayesian model averaging for quantifying predictive uncertainty?

Is there a scalable alternative to Bayesian model averaging for quantifying predictive uncertainty? Yes!

Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles

Balaji Lakshminarayanan **Alexander Pritzel** **Charles Blundell**
DeepMind
{balajiln,apritzel,cblundell}@google.com

Our contribution: simple yet powerful baseline

Probabilistic, but non-Bayesian, baseline

Our contribution: simple yet powerful baseline

Probabilistic, but non-Bayesian, baseline

- Performs well on evaluation metrics
- Simple to implement (minimal changes to baseline)
- Scalable to large datasets (e.g. ImageNet)

Our contribution: simple yet powerful baseline

Probabilistic, but non-Bayesian, baseline

- Performs well on evaluation metrics
- Simple to implement (minimal changes to baseline)
- Scalable to large datasets (e.g. ImageNet)
- Robust:
  - Works for different output types (classification/regression)
  - Works for different architectures

A Simple Recipe for Uncertainty Estimation

1. Let **neural network parametrize a distribution** $p_{\boldsymbol{\theta}}(y|\mathbf{x})$.
   - Classification: softmax parametrizes discrete distribution
   - Regression: Gaussian with mean $\mu_{\boldsymbol{\theta}}(\mathbf{x})$ & var $\sigma^2_{\boldsymbol{\theta}}(\mathbf{x})$

A Simple Recipe for Uncertainty Estimation

1. Let **neural network parametrize a distribution** $p_{\boldsymbol{\theta}}(y|\mathbf{x})$.
   - Classification: softmax parametrizes discrete distribution
   - Regression: Gaussian with mean $\mu_{\boldsymbol{\theta}}(\mathbf{x})$ & var $\sigma^2_{\boldsymbol{\theta}}(\mathbf{x})$
2. Use a **proper scoring rule** as training criterion.
   - Classification: cross entropy loss
   - Regression: Gaussian likelihood mean $\mu_{\boldsymbol{\theta}}(\mathbf{x})$ & var $\sigma^2_{\boldsymbol{\theta}}(\mathbf{x})$

A Simple Recipe for Uncertainty Estimation

1. Let **neural network parametrize a distribution** $p_{\boldsymbol{\theta}}(y|\mathbf{x})$.
   - Classification: softmax parametrizes discrete distribution
   - Regression: Gaussian with mean $\mu_{\boldsymbol{\theta}}(\mathbf{x})$ & var $\sigma^2_{\boldsymbol{\theta}}(\mathbf{x})$
2. Use a **proper scoring rule** as training criterion.
   - Classification: cross entropy loss
   - Regression: Gaussian likelihood mean $\mu_{\boldsymbol{\theta}}(\mathbf{x})$ & var $\sigma^2_{\boldsymbol{\theta}}(\mathbf{x})$
3. (Optional) Augment with **adversarial training**
   - Augment $(\mathbf{x} + \Delta\mathbf{x}, y)$ where $\Delta\mathbf{x} = -\epsilon \, \mathrm{sign}\Big(\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(y|\mathbf{x})\Big)$
   - Encourages $p(y|\mathbf{x})$ to be similar to $p(y|\mathbf{x} + \Delta\mathbf{x})$

A Simple Recipe for Uncertainty Estimation

1. Let **neural network parametrize a distribution** $p_{\boldsymbol{\theta}}(y|\mathbf{x})$.
   - Classification: softmax parametrizes discrete distribution
   - Regression: Gaussian with mean $\mu_{\boldsymbol{\theta}}(\mathbf{x})$ & var $\sigma^2_{\boldsymbol{\theta}}(\mathbf{x})$
2. Use a **proper scoring rule** as training criterion.
   - Classification: cross entropy loss
   - Regression: Gaussian likelihood mean $\mu_{\boldsymbol{\theta}}(\mathbf{x})$ & var $\sigma^2_{\boldsymbol{\theta}}(\mathbf{x})$
3. (Optional) Augment with **adversarial training**
   - Augment $(\mathbf{x} + \Delta\mathbf{x}, y)$ where $\Delta\mathbf{x} = -\epsilon \, \text{sign}\left(\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(y|\mathbf{x})\right)$
   - Encourages $p(y|\mathbf{x})$ to be similar to $p(y|\mathbf{x} + \Delta\mathbf{x})$
4. Train an **ensemble of *M* networks** with random initialization

A Simple Recipe for Uncertainty Estimation

1. Let **neural network parametrize a distribution** $p_{\boldsymbol{\theta}}(y|\mathbf{x})$.
   - Classification: softmax parametrizes discrete distribution
   - Regression: Gaussian with mean $\mu_{\boldsymbol{\theta}}(\mathbf{x})$ & var $\sigma^2_{\boldsymbol{\theta}}(\mathbf{x})$
2. Use a **proper scoring rule** as training criterion.
   - Classification: cross entropy loss
   - Regression: Gaussian likelihood mean $\mu_{\boldsymbol{\theta}}(\mathbf{x})$ & var $\sigma^2_{\boldsymbol{\theta}}(\mathbf{x})$
3. (Optional) Augment with **adversarial training**
   - Augment $(\mathbf{x} + \Delta\mathbf{x}, y)$ where $\Delta\mathbf{x} = -\epsilon \operatorname{sign}\left(\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(y|\mathbf{x})\right)$
   - Encourages $p(y|\mathbf{x})$ to be similar to $p(y|\mathbf{x} + \Delta\mathbf{x})$
4. Train an **ensemble of M networks** with random initialization
5. Combine predictions at test time

$$p(y|\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} p_{\boldsymbol{\theta}_m}(y|\mathbf{x}, \boldsymbol{\theta}_m)$$

A Simple Recipe for Uncertainty Estimation

1. Let **neural network parametrize a distribution** $p_{\boldsymbol{\theta}}(y|\mathbf{x})$.
   - Classification: softmax parametrizes discrete distribution
   - Regression: Gaussian with mean $\mu_{\boldsymbol{\theta}}(\mathbf{x})$ & var $\sigma^2_{\boldsymbol{\theta}}(\mathbf{x})$
2. Use a **proper scoring rule** as training criterion.
   - Classification: cross entropy loss
   - Regression: Gaussian likelihood mean $\mu_{\boldsymbol{\theta}}(\mathbf{x})$ & var $\sigma^2_{\boldsymbol{\theta}}(\mathbf{x})$
3. (Optional) Augment with **adversarial training**
   - Augment $(\mathbf{x} + \Delta\mathbf{x}, y)$ where $\Delta\mathbf{x} = -\epsilon \text{ sign}\Big(\nabla_{\mathbf{x}} \ \log p_{\boldsymbol{\theta}}(y|\mathbf{x})\Big)$
   - Encourages $p(y|\mathbf{x})$ to be similar to $p(y|\mathbf{x} + \Delta\mathbf{x})$
4. Train an **ensemble of M networks** with random initialization
5. Combine predictions at test time

$$p(y|\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} p_{\boldsymbol{\theta}_m}(y|\mathbf{x}, \boldsymbol{\theta}_m)$$

Model combination & not Bayesian Model Averaging

BMA vs Model combination[2]

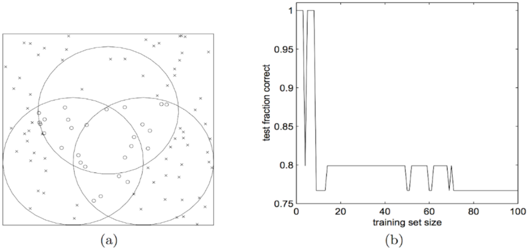
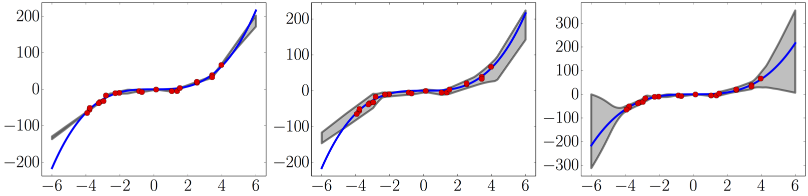


Figure 1: (a) A classification problem where points in 2D are labeled with 'x' or 'o'. The optimal solution is to label 'o' if a point is in at least two circles, corresponding to a uniform vote between the circles. (b) The test-set accuracy of BMA, as a function of training set size. BMA always focuses on the topmost circle, even though the other two circles have nearly the same accuracy.

- BMA does **soft model selection**: BMA converges to single model in infinite data limit.
- Ensembles do model combination. Hypothesis space is richer (as it is an additive model).

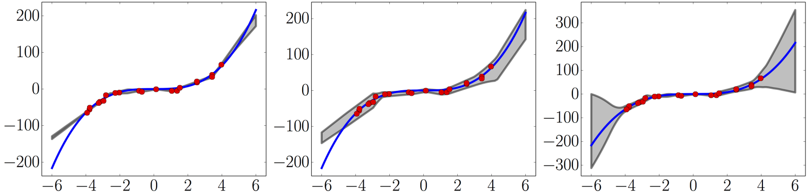
[2] Example from *Bayesian Model Averaging Is Not Model Combination* (Minka, 2001)

Results on a toy regression task



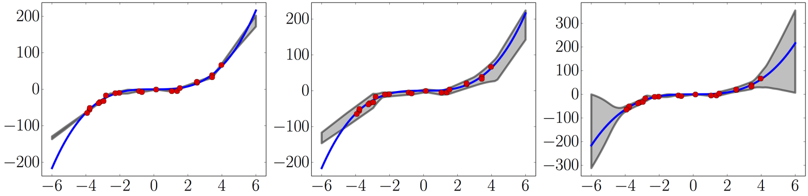
- Left plot: **non-probabilistic network**, use empirical variance between 5 networks as uncertainty

Results on a toy regression task



- Left plot: **non-probabilistic network**, use empirical variance between 5 networks as uncertainty
- Middle plot: single probabilistic network

Results on a toy regression task



- Left plot: **non-probabilistic network**, use empirical variance between 5 networks as uncertainty
- Middle plot: single probabilistic network
- Right plot: **ensemble of 5 probabilistic networks**.

Results on UCI regression benchmark datasets

Datasets	RMSE			NLL		
	PBP	MC-dropout	Deep Ensembles	PBP	MC-dropout	Deep Ensembles
Boston housing	**3.01 ± 0.18**	**2.97 ± 0.85**	**3.28 ± 1.00**	**2.57 ± 0.09**	**2.46 ± 0.25**	**2.41 ± 0.25**
Concrete	**5.67 ± 0.09**	**5.23 ± 0.53**	**6.03 ± 0.58**	**3.16 ± 0.02**	**3.04 ± 0.09**	**3.06 ± 0.18**
Energy	**1.80 ± 0.05**	**1.66 ± 0.19**	**2.09 ± 0.29**	2.04 ± 0.02	1.99 ± 0.09	**1.38 ± 0.22**
Kin8nm	0.10 ± 0.00	0.10 ± 0.00	**0.09 ± 0.00**	-0.90 ± 0.01	-0.95 ± 0.03	**-1.20 ± 0.02**
Naval propulsion plant	0.01 ± 0.00	0.01 ± 0.00	**0.00 ± 0.00**	-3.73 ± 0.01	-3.80 ± 0.05	**-5.63 ± 0.05**
Power plant	**4.12 ± 0.03**	**4.02 ± 0.18**	**4.11 ± 0.17**	2.84 ± 0.01	**2.80 ± 0.05**	**2.79 ± 0.04**
Protein	4.73 ± 0.01	**4.36 ± 0.04**	4.71 ± 0.06	2.97 ± 0.00	2.89 ± 0.01	**2.83 ± 0.02**
Wine	**0.64 ± 0.01**	**0.62 ± 0.04**	**0.64 ± 0.04**	0.97 ± 0.01	**0.93 ± 0.06**	**0.94 ± 0.12**
Yacht	**1.02 ± 0.05**	**1.11 ± 0.38**	**1.58 ± 0.48**	1.63 ± 0.02	1.55 ± 0.12	**1.18 ± 0.21**
Year Prediction MSD	8.88 ± NA	**8.85 ± NA**	8.89 ± NA	3.60 ± NA	3.59 ± NA	**3.35 ± NA**

- Our method achieves better NLL, but slightly worse RMSE in some cases
- Even though non-Bayesian, our method is competitive with probabilistic backpropagation (PBP) and MC-Dropout

Calibration results on Year Prediction MSD

Probabilistic networks (left) are much better calibrated than non-probabilistic networks (right).

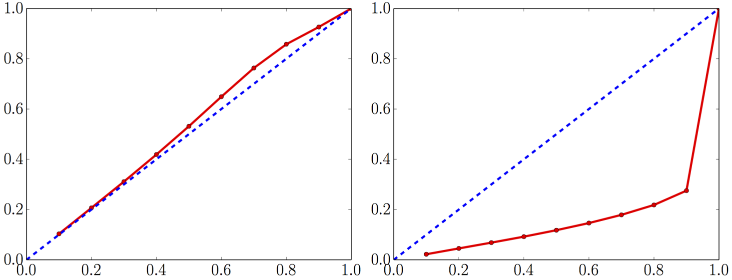
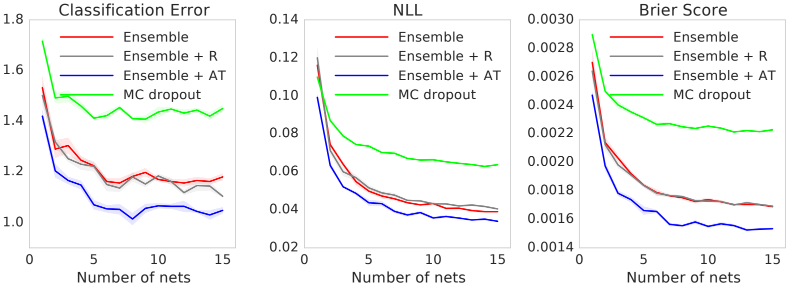


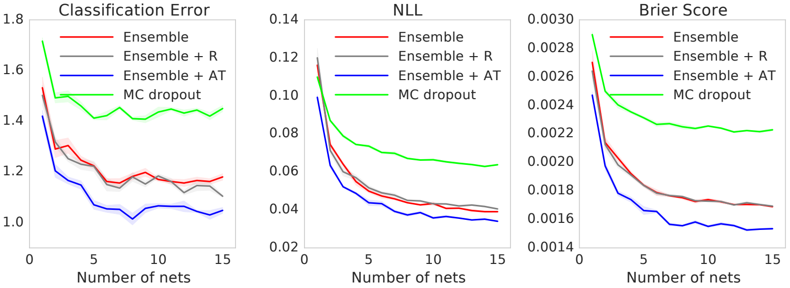
Figure: *x*-axis denotes the expected fraction and *y*-axis denotes the observed fraction; ideal output is the dashed blue line. Above diagonal = under-confidence, below diagonal = over-confidence.

Classification Results on MNIST using MLP



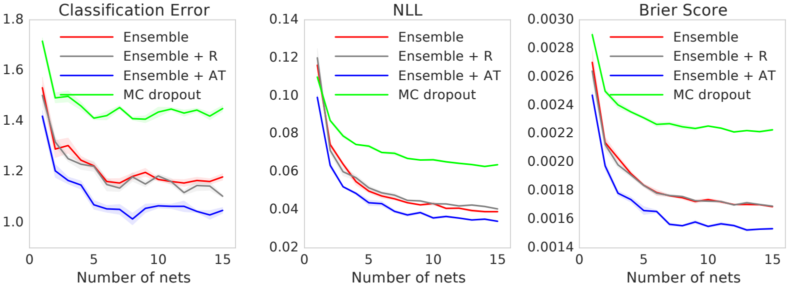
- **Ensembles lead to better predictive uncertainty**

Classification Results on MNIST using MLP



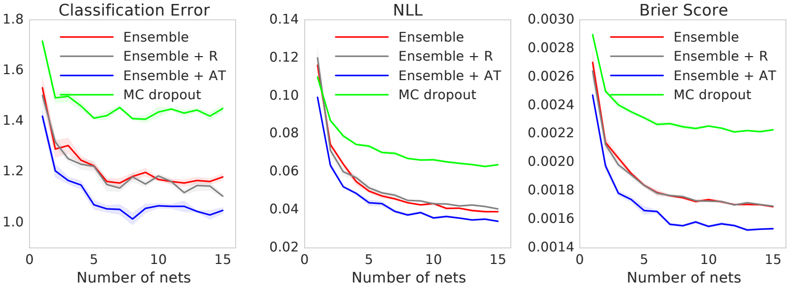
- **Ensembles lead to better predictive uncertainty**
- Adversarial training leads to further improvements

Classification Results on MNIST using MLP



- **Ensembles lead to better predictive uncertainty**
- Adversarial training leads to further improvements
- Similar results on SVHN using convolutional nets

Classification Results on MNIST using MLP



- **Ensembles lead to better predictive uncertainty**
- Adversarial training leads to further improvements
- Similar results on SVHN using convolutional nets
- We also show **results on ImageNet to illustrate scalability**

Evaluating predictive uncertainty on OOD

- Goal: check if the methods are more uncertain while testing on out-of-distribution (OOD) dataset.

Evaluating predictive uncertainty on OOD

- Goal: check if the methods are more uncertain while testing on out-of-distribution (OOD) dataset.
- Setup:
  - Train on MNIST
  - Evaluate on known test set (MNIST) and unknown test set (NotMNIST) (both 28 x 28 gray-scale images)

Evaluating predictive uncertainty on OOD

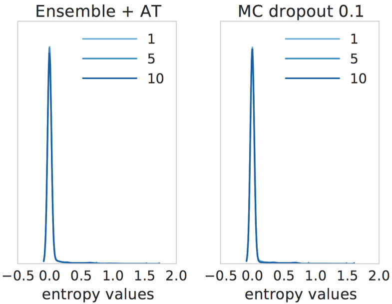
- Goal: check if the methods are more uncertain while testing on out-of-distribution (OOD) dataset.
- Setup:
  - Train on MNIST
  - Evaluate on known test set (MNIST) and unknown test set (NotMNIST) (both 28 x 28 gray-scale images)
- Also trained / tested on different datasets:
  - Train on SVHN / Test on CIFAR (both 32 x 32 x 3 images)
  - ImageNet: train on dog categories and test on non-dog categories

Predictive entropy on known & unknown inputs

Train: MNIST. **Test**: MNIST + NotMNIST (out-of-distribution)

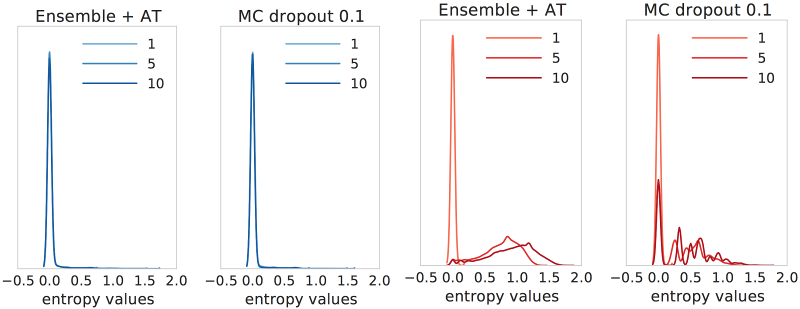
Predictive entropy on known & unknown inputs

Train: MNIST. **Test**: MNIST + NotMNIST (out-of-distribution)



Predictive entropy on known & unknown inputs

Train: MNIST. **Test**: MNIST + NotMNIST (out-of-distribution)



Single network & MC-dropout can produce **overconfident wrong predictions**, whereas **deep ensembles are more robust.**
Similar results on SVHN-CIFAR and ImageNet (dogs vs no-dogs).

Accuracy Vs Confidence

Model abstains from making prediction when confidence $< \tau$

Evaluate test accuracy only on examples where $\max_y p(y|\mathbf{x}) \geq \tau$

Accuracy Vs Confidence

Model abstains from making prediction when confidence $< \tau$

Evaluate test accuracy only on examples where $\max_y p(y|\mathbf{x}) \geq \tau$

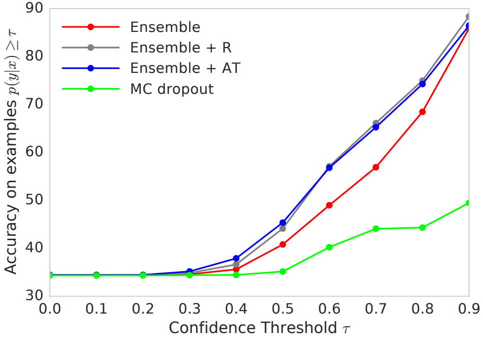
Train: MNIST. **Test**: MNIST + NotMNIST (out-of-distribution)

Accuracy Vs Confidence

Model abstains from making prediction when confidence $< \tau$

Evaluate test accuracy only on examples where $\max_y p(y|\mathbf{x}) \geq \tau$

Train: MNIST. **Test**: MNIST + NotMNIST (out-of-distribution)



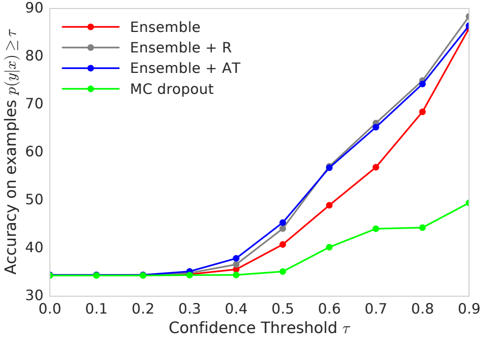
- MC-dropout can produce **overconfident wrong predictions**, whereas **deep ensembles are significantly more robust.**

Accuracy Vs Confidence

Model abstains from making prediction when confidence $< \tau$

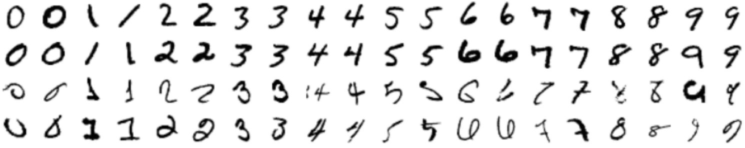
Evaluate test accuracy only on examples where $\max_y p(y|\mathbf{x}) \geq \tau$

Train: MNIST. **Test**: MNIST + NotMNIST (out-of-distribution)



- MC-dropout can produce **overconfident wrong predictions**, whereas **deep ensembles are significantly more robust.**
- Similar results on ImageNet (dogs vs no-dogs)

Qualitatively evaluating predictive uncertainty



- Top two rows: examples with lowest disagreement
- Bottom two rows: examples with highest disagreement

Why would this work?

- Modeling distribution $p_\theta(y|\mathbf{x})$ captures inherent ambiguity (aleatoric uncertainty).

Why would this work?

- Modeling distribution $p_\theta(y|\mathbf{x})$ captures inherent ambiguity (aleatoric uncertainty).

Why would this work?

- Modeling distribution $p_\theta(y|\mathbf{x})$ captures inherent ambiguity (aleatoric uncertainty).
- Ensemble approximates epistemic uncertainty
  - Training on bootstrap samples has theoretical justification
  - In practice, using entire dataset works better.

Why would this work?

- Modeling distribution $p_\theta(y|\mathbf{x})$ captures inherent ambiguity (aleatoric uncertainty).
- Ensemble approximates epistemic uncertainty
  - Training on bootstrap samples has theoretical justification
  - In practice, using entire dataset works better.
- Interesting similarities to ensembles of decision trees
  - Breiman's random forests [2] used bagging
  - Later work on Extremely Randomized Trees found bagging to be unnecessary if there was sufficient randomization [4]
  - (Non-Bayesian) Ensembles of probabilistic decision trees can give good uncertainty estimates in practice [5]

nature medicine

ARTICLES
https://doi.org/10.1038/s41591-018-0107-6

Clinically applicable deep learning for diagnosis and referral in retinal disease

Jeffrey De Fauw[1], Joseph R. Ledsam[1], Bernardino Romera-Paredes[1], Stanislav Nikolov[1], Nenad Tomasev[1], Sam Blackwell[1], Harry Askham[1], Xavier Glorot[1], Brendan O'Donoghue[1], Daniel Visentin[1], George van den Driessche[1], Balaji Lakshminarayanan[1], Clemens Meyer[1], Faith Mackinder[1], Simon Bouton[1], Kareem Ayoub[1], Reena Chopra[2], Dominic King[1], Alan Karthikesalingam[1], Cían O. Hughes[1,3], Rosalind Raine[3], Julian Hughes[2], Dawn A. Sim[2], Catherine Egan[2], Adnan Tufail[2], Hugh Montgomery[3], Demis Hassabis[1], Geraint Rees[3], Trevor Back[1], Peng T. Khaw[2], Mustafa Suleyman[1], Julien Cornebise[1,3,4], Pearse A. Keane[2,4*] and Olaf Ronneberger[1,4*]

The volume and complexity of diagnostic imaging is increasing at a pace faster than the availability of human expertise to interpret it. Artificial intelligence has shown great promise in classifying two-dimensional photographs of some common diseases and typically relies on databases of millions of annotated images. Until now, the challenge of reaching the performance of expert clinicians in a real-world clinical pathway with three-dimensional diagnostic scans has remained unsolved. Here, we apply a novel deep learning architecture to a clinically heterogeneous set of three-dimensional optical coherence tomography scans from patients referred to a major eye hospital. We demonstrate performance in making a referral recommendation that reaches or exceeds that of experts on a range of sight-threatening retinal diseases after training on only 14,884 scans. Moreover, we demonstrate that the tissue segmentations produced by our architecture act as a device-independent representation; referral accuracy is maintained when using tissue segmentations from a different type of device. Our work removes previous barriers to wider clinical use without prohibitive training data requirements across multiple pathologies in a real-world setting.

Medical imaging is expanding globally at an unprecedented rate[1,2], leading to an ever-expanding quantity of data that requires human expertise and judgement to interpret and triage. In many clinical specialities there is a relative shortage of this expertise to provide timely diagnosis and referral. For example, in ophthalmology, the widespread availability of optical coherence OCT has shown promise in resolving some of these criteria in isolation, but has not yet shown clinical applicability by resolving all three.

Results

Clinical application and AI architecture. We developed our architecture in the challenging context of OCT imaging for oph-

Take home message

- Non-Bayesian, Probabilistic solutions can be surprisingly effective at estimating predictive uncertainty

Take home message

- Non-Bayesian, Probabilistic solutions can be surprisingly effective at estimating predictive uncertainty
- Strong non-Bayesian baselines are valuable to understand the limitations
  - Better ways to specify priors
  - Better ways to improve approximate posteriors
- Lots of other promising non-Bayesian solutions
  - Temperature scaling
  - ODIN

Take home message

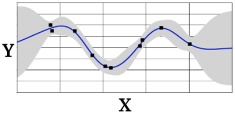
- Non-Bayesian, Probabilistic solutions can be surprisingly effective at estimating predictive uncertainty
- Strong non-Bayesian baselines are valuable to understand the limitations
  - Better ways to specify priors
  - Better ways to improve approximate posteriors
- Lots of other promising non-Bayesian solutions
  - Temperature scaling
  - ODIN
- Combining Bayesian and non-Bayesian solutions can get the best of both worlds

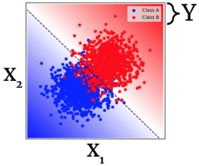
Published as a conference paper at ICLR 2019

Do Deep Generative Models Know What They Don't Know?

Eric Nalisnick*†, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, Balaji Lakshminarayanan*
DeepMind

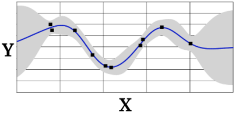
So far: Discriminative models

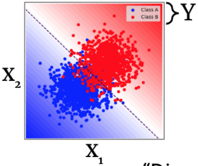




$$p(\mathbf{y}|\mathbf{x})$$

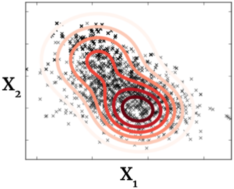
Discriminative vs Generative models





$$p(\mathbf{y}|\mathbf{x})$$

"Discriminative" Model



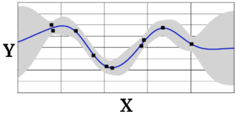
$$p(\mathbf{x})$$

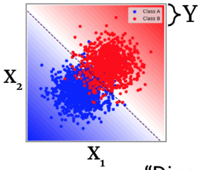
"Generative" Model

- $p(y|x)$ is typically accurate when $x \sim p_{train}(x)$, but can make overconfident errors when asked to predict on OOD

[3] *Novelty Detection and Neural Network Validation* (Bishop, 1994)

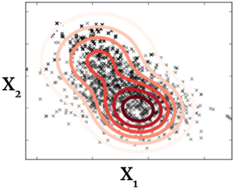
Discriminative vs Generative models





$$p(\mathbf{y}|\mathbf{x})$$

"Discriminative" Model



$$p(\mathbf{x})$$

"Generative" Model

- $p(y|x)$ is typically accurate when $x \sim p_{train}(x)$, but can make overconfident errors when asked to predict on OOD
- Use generative model to decide when to trust $p(y|x)$ [1][3]

[3] *Novelty Detection and Neural Network Validation* (Bishop, 1994)

Inputs Unlike Training Data

$$\text{if } p(\boldsymbol{x}^*; \boldsymbol{\phi}) < \tau,$$
$$\text{then } \texttt{reject } \boldsymbol{x}^*$$

Use p(**X**) model to reject inputs with density below some threshold [Bishop, 1994].

AABI workshop, NeurIPS 2017[4]

Panel Discussion, Advances in Approximate Bayesian Inference (AABI) workshop

[4] https://www.youtube.com/watch?v=x1UByHT60mQ&feature=youtu.be&t=46m2s

AABI workshop, NeurIPS 2017

ZOUBIN: [The Bishop (1994) procedure] should be built into the software.

AABI workshop, NeurIPS 2017

ZOUBIN: [The Bishop (1994) procedure] should be built into the software.

MODERATOR: Isn't that hard?

AABI workshop, NeurIPS 2017

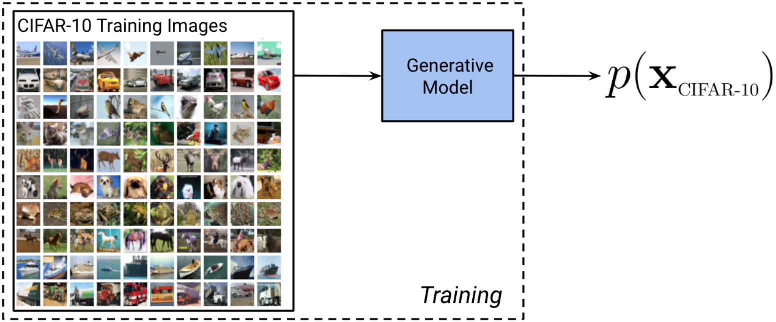
ZOUBIN: [The Bishop (1994) procedure] should be built into the software.

MODERATOR: Isn't that hard?

ZOUBIN: If you stick a picture of a chicken into an MNIST classifier, it should tell you it's neither a seven nor a one.

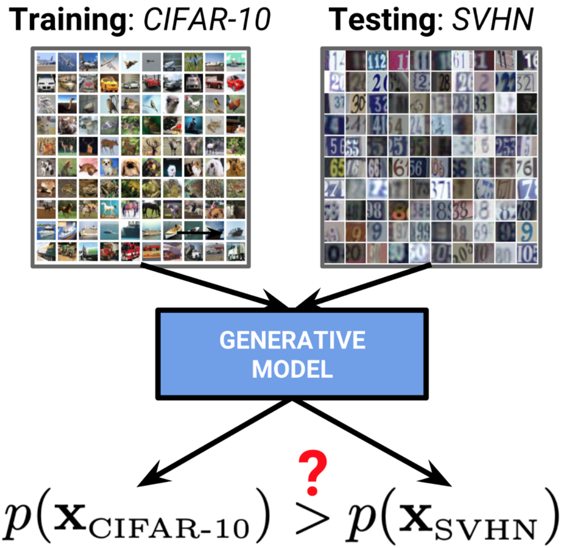
[*AUDIENCE LAUGHS*]

Generative models for CIFAR



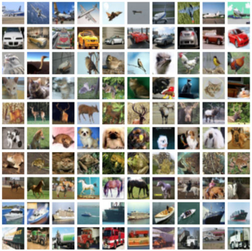
Deep generative models where density $p(x)$ can be computed:
Flows, Auto-regressive models, VAEs (lower bound)

Training on CIFAR and Testing on SVHN (OOD)



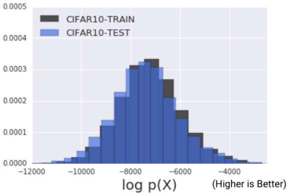
Training a Flow-Based Model on CIFAR-10

CIFAR-10 Training Images



	Bits Per Dimension (NLL / # dims / log 2)
CIFAR10-Train	3.386
CIFAR10-Test	3.464

(Lower is Better)

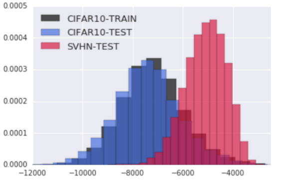


Training a Flow-Based Model on CIFAR-10

SVHN Test Images

Bits Per Dimension (NLL / # dims / log 2)	
CIFAR10-Train	3.386
CIFAR10-Test	3.464
SVHN-Test	**2.389**

(Lower is Better)



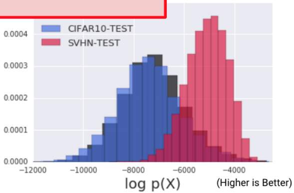
Training a Flow-Based Model on CIFAR-10

SVHN Test Images

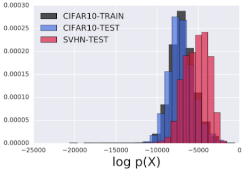
	Bits Per Dimension (NLL / # dims / log 2)
CIFAR10-Train	3.386
[illegible]	3.464
[illegible]	**2.389**

(Lower is Better)

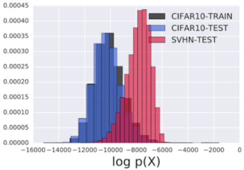
Big Problem!



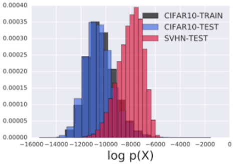
Phenomenon holds for VAEs and PixelCNN too



(a) PixelCNN

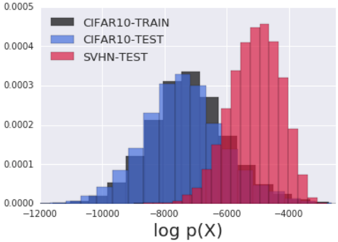


(b) VAE with RNVP as encoder

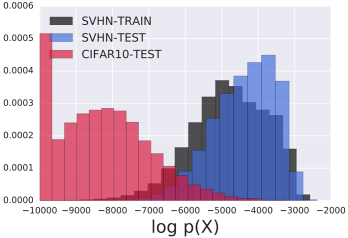


(c) VAE conv-categorical likelihood

The phenomenon is asymmetric w.r.t. datasets

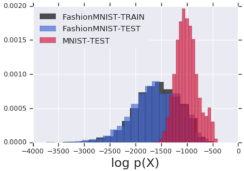


CIFAR-10 vs **SVHN**

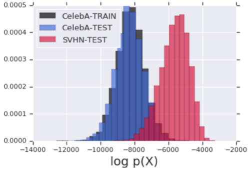


SVHN vs **CIFAR-10**

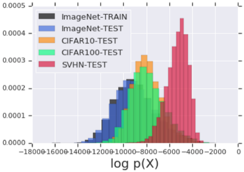
Additional OOD dataset pairs



FashionMNIST vs **MNIST**

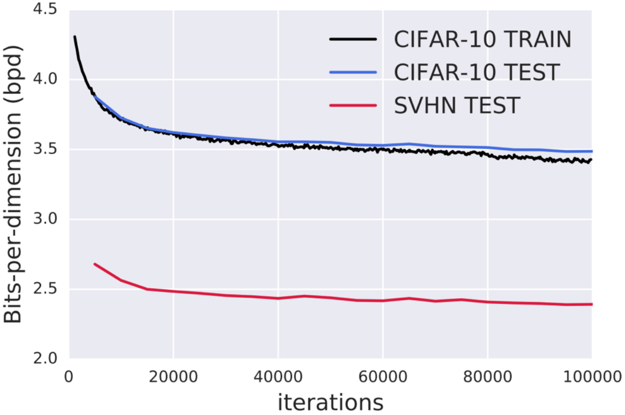


CelebA vs **SVHN**



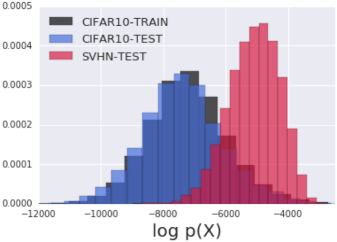
ImageNet vs **CIFAR-10**
vs **SVHN**

Not caused by overfitting: Early stopping does not help

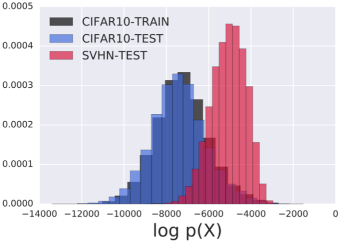


During Optimization

Ensembling does not fix the problem either



CIFAR-10 vs SVHN
1 Glow



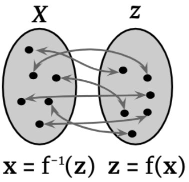
CIFAR-10 vs SVHN
Ensemble of 10 Glows

Digging deeper into flows

Flows: one slide summary

Define *Z* by a transformation of another variable *X*: $Z = f(X)$

f(**x**) is a *bijection (invertible 1:1 mapping)*



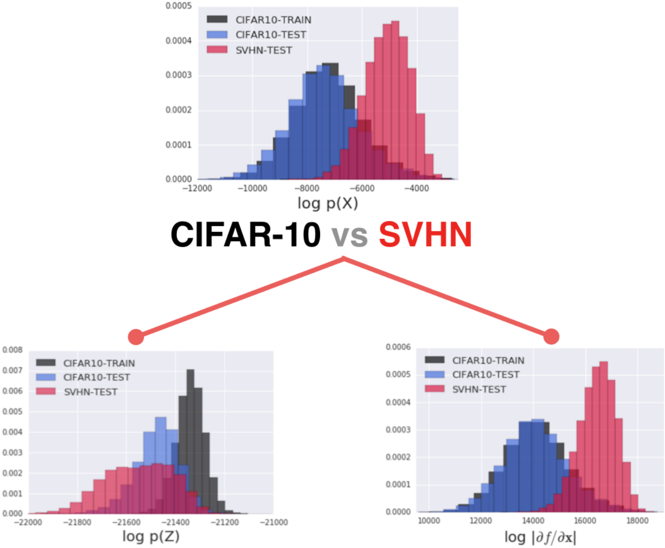
Change of Variables Formula (X → Z):

$$p_z(f(X)) \left| \frac{df(X)}{dX} \right| = p(X)$$

Use simple p_z distribution (e.g. standard normal)

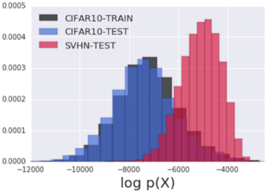
Use f such that the Jacobian df/dx is easy to compute

Decomposition of likelihood for flow models



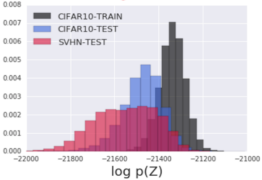
Distribution Term **Volume Term**

Decomposition of likelihood for flow models

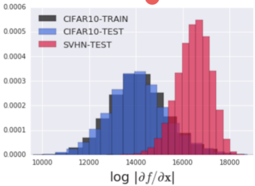


CIFAR-10 vs SVHN

- Looks to be the cause of the phenomenon



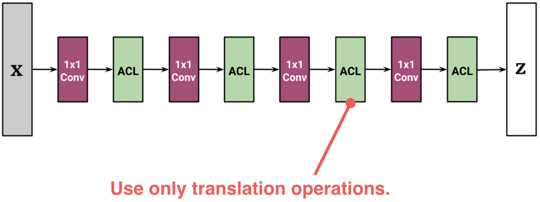
Distribution Term



Volume Term

Is the log volume term the culprit?

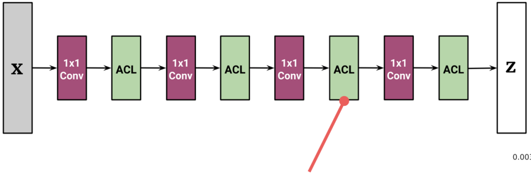
We define a sub-class we term *constant-volume* (w.r.t. input) flows.



To isolate the effect of the volume term, we define **constant-volume (w.r.t. input) flows**.

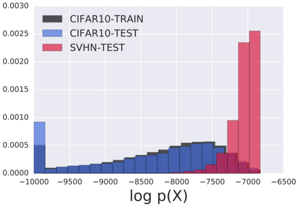
Is the log volume term the culprit? No.

We define a sub-class we term *constant-volume* (w.r.t. input) flows.



Use only translation operations.

CIFAR-10 vs SVHN



Analysis of Constant Volume GLOW models

Analysis of Constant Volume GLOW models

Mathematical characterization:

$$0 < \underbrace{\mathbb{E}_q}_{\text{Non-Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \underbrace{\mathbb{E}_{p^*}}_{\text{Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

Analysis of Constant Volume GLOW models

Mathematical characterization:

$$0 < \underbrace{\mathbb{E}_q}_{\text{Non-Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \underbrace{\mathbb{E}_{p^*}}_{\text{Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

$$\approx \frac{1}{2}\mathrm{Tr}\left\{\left[\underbrace{\nabla^2_{\boldsymbol{x}_0}\log p_z(f(\boldsymbol{x}_0;\boldsymbol{\phi})) + \nabla^2_{\boldsymbol{x}_0}\log\left|\frac{\partial \boldsymbol{f}_\phi}{\partial \boldsymbol{x}_0}\right|}_{\text{Change-of-Variable Terms}}\right](\underbrace{\boldsymbol{\Sigma}_q}_{\text{Second Moment of Non-Training Distribution}} - \overbrace{\boldsymbol{\Sigma}_{p^*}}^{\text{Second Moment of Training Distribution}})\right\}$$

Analysis of Constant Volume GLOW models

Mathematical characterization:

$$0 < \underbrace{\mathbb{E}_q}_{\substack{\text{Non-Training}\\\text{Distribution}}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \underbrace{\mathbb{E}_{p^*}}_{\substack{\text{Training}\\\text{Distribution}}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

$$\approx \frac{1}{2}\operatorname{Tr}\left\{\left[\underbrace{\nabla^2_{\boldsymbol{x}_0}\log p_z(f(\boldsymbol{x}_0;\boldsymbol{\phi})) + \cancel{\nabla^2_{\boldsymbol{x}_0}\log\left|\frac{\partial \boldsymbol{f}_{\boldsymbol{\phi}}}{\partial \boldsymbol{x}_0}\right|}}_{\text{Change-of-Variable Terms}}\right]\left(\underbrace{\boldsymbol{\Sigma}_q}_{\substack{\text{Second Moment}\\\text{of Non-Training}\\\text{Distribution}}} - \overbrace{\boldsymbol{\Sigma}_{p^*}}^{\substack{\text{Second Moment}\\\text{of Training}\\\text{Distribution}}}\right)\right\}$$

Analysis of Constant Volume GLOW models

Plugging in the CV-Glow transform:

$$\mathrm{Tr}\left\{\left[\nabla^2_{\boldsymbol{x}_0} \log p(\boldsymbol{x}_0;\boldsymbol{\theta})\right](\boldsymbol{\Sigma}_q - \boldsymbol{\Sigma}_{p^*})\right\}$$

$$= \frac{\partial^2}{\partial z^2}\log p(\boldsymbol{z};\boldsymbol{\psi}) \sum_{c=1}^{C}\left(\prod_{k=1}^{K}\sum_{j=1}^{C}\underbrace{u_{k,c,j}}_{\text{1x1 Conv. Params}}\right)^2 \sum_{h,w}\left(\overbrace{\sigma^2_{q,h,w,c}}^{\text{Second Moment of Non-Training Distribution}} - \overbrace{\sigma^2_{p^*,h,w,c}}^{\text{Second Moment of Training Distribution}}\right)$$

Analysis of Constant Volume GLOW models

Plugging in the CV-Glow transform:

$$\operatorname{Tr}\left\{\left[\nabla^2_{\boldsymbol{x}_0}\log p(\boldsymbol{x}_0;\boldsymbol{\theta})\right]\left(\boldsymbol{\Sigma}_q-\boldsymbol{\Sigma}_{p^*}\right)\right\}$$

$$=\frac{\partial^2}{\partial z^2}\log p(\boldsymbol{z};\boldsymbol{\psi})\sum_{c=1}^{C}\left(\prod_{k=1}^{K}\sum_{j=1}^{C}u_{k,c,j}\right)^2\sum_{h,w}\left(\sigma^2_{q,h,w,c}-\sigma^2_{p^*,h,w,c}\right)$$

Second Moment of Non-Training Distribution

Second Moment of Training Distribution

1x1 Conv. Params

Sums over channel dimensions

Product over steps in flow

Sum over spatial dimensions

Analysis of Constant Volume GLOW models

Plugging in the CV-Glow transform:

$$\mathrm{Tr}\left\{\left[\nabla^2_{\boldsymbol{x}_0} \log p(\boldsymbol{x}_0;\boldsymbol{\theta})\right](\boldsymbol{\Sigma}_q - \boldsymbol{\Sigma}_{p^*})\right\}$$

$$= \underbrace{\frac{\partial^2}{\partial z^2}\log p(\boldsymbol{z};\boldsymbol{\psi})}_{<0 \text{ for all log-concave densities (e.g. Gaussian)}} \sum_{c=1}^{C} \underbrace{\left(\prod_{k=1}^{K}\sum_{j=1}^{C} u_{k,c,j}\right)^2}_{\text{Non-negative due to square}} \sum_{h,w}(\underbrace{\sigma^2_{q,h,w,c}}_{\text{Second Moment of Non-Training Distribution}} - \underbrace{\sigma^2_{p^*,h,w,c}}_{\text{Second Moment of Training Distribution}})$$

< 0 for all log-concave densities (e.g. Gaussian)

Non-negative due to square

Second Moment of Non-Training Distribution

Second Moment of Training Distribution

Analysis of Constant Volume GLOW models

Plugging in the CV-Glow transform:

$$\mathrm{Tr}\left\{\left[\nabla^2_{\boldsymbol{x}_0}\log p(\boldsymbol{x}_0;\boldsymbol{\theta})\right]\left(\boldsymbol{\Sigma}_q - \boldsymbol{\Sigma}_{p^*}\right)\right\}$$

$$= \frac{\partial^2}{\partial z^2}\log p(\boldsymbol{z};\boldsymbol{\psi})\sum_{c=1}^{C}\left(\prod_{k=1}^{K}\sum_{j=1}^{C} u_{k,c,j}\right)^2 \sum_{h,w}\left(\sigma^2_{q,h,w,c} - \sigma^2_{p^*,h,w,c}\right)$$

< 0 for all log-concave densities (e.g. Gaussian)

Non-negative due to square

Second Moment of Non-Training Distribution

Second Moment of Training Distribution

Sign boils down to difference in moments. Speaks to asymmetric behavior.

Analysis of Constant Volume GLOW models

Plugging in the CIFAR-10 and SVHN statistics:

$$\mathbb{E}_{\text{SVHN}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \mathbb{E}_{\text{CIFAR10}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

$$\approx \frac{1}{2\sigma_{\psi}^2}\left[\alpha_1^2 \cdot 12.3 + \alpha_2^2 \cdot 6.5 + \alpha_3^2 \cdot 14.5\right] \geq 0 \quad \text{where} \;\; \alpha_c = \prod_{k=1}^{K}\sum_{j=1}^{C} u_{k,c,j}$$

Differences in variances in the three spatial dimensions

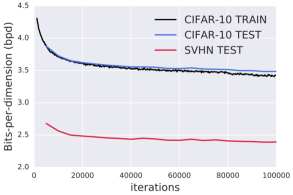
Analysis of Constant Volume GLOW models

Plugging in the CIFAR-10 and SVHN statistics:

$$\mathbb{E}_{\text{SVHN}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \mathbb{E}_{\text{CIFAR10}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

$$\approx \frac{1}{2\sigma_\psi^2}\left[\alpha_1^2 \cdot 12.3 + \alpha_2^2 \cdot 6.5 + \alpha_3^2 \cdot 14.5\right] \geq 0 \quad \text{where} \quad \alpha_c = \prod_{k=1}^{K} \sum_{j=1}^{C} u_{k,c,j}$$

Differences in variances in the three spatial dimensions



The expression will be non-negative **for any** parameter setting of the CV flow….

Analysis of Constant Volume GLOW models

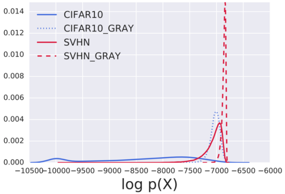
Plugging in the CIFAR-10 and SVHN statistics:

$$\mathbb{E}_{\text{SVHN}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \mathbb{E}_{\text{CIFAR10}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

$$\approx \frac{1}{2\sigma_{\psi}^2}\left[\alpha_1^2 \cdot 12.3 + \alpha_2^2 \cdot 6.5 + \alpha_3^2 \cdot 14.5\right] \geq 0 \quad \text{where} \;\; \alpha_c = \prod_{k=1}^{K}\sum_{j=1}^{C} u_{k,c,j}$$

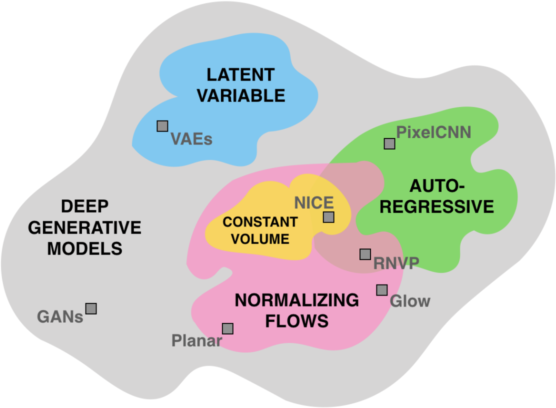
Differences in variances in the three spatial dimensions

This also means that we can manipulate the relative log likelihoods just by changing the variance of the data. For natural images, this amounts to **graying**…



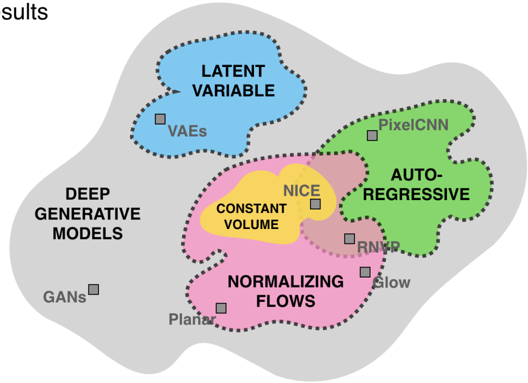
One weird trick to increase likelihoods: grayscale images!

Summary of Results

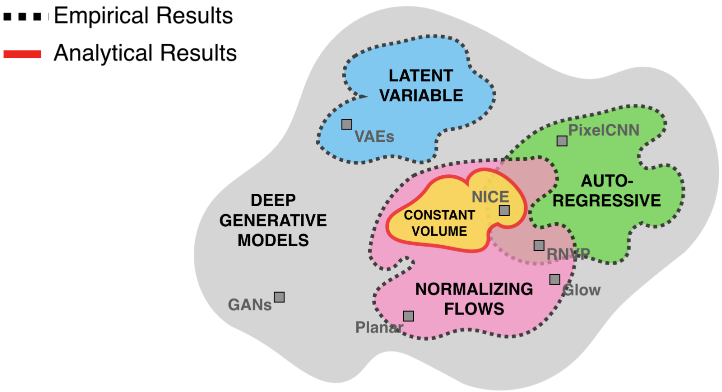


Summary of Results

▪ ▪ ▪ Empirical Results



Summary of Results



Take home message

- Deep generative models are attractive but have problems detecting out-of-distribution data.

Take home message

- Deep generative models are attractive but have problems detecting out-of-distribution data.
- For flow-based models, the phenomenon can be explained through the relative variances of the different input distributions
  - Grayscale images
  - Constant images

Take home message

- Deep generative models are attractive but have problems detecting out-of-distribution data.
- For flow-based models, the phenomenon can be explained through the relative variances of the different input distributions
  - Grayscale images
  - Constant images
- Be cautious when using density estimates from deep generative models as proxy for “similarity” to training data
  - Novelty detection
  - Anomaly detection

Papers available on my webpage (link)

- *Simple and scalable predictive uncertainty estimation using deep ensembles*, NeurIPS, 2017 [6]
- *Clinically applicable deep learning for diagnosis and referral in retinal disease*, Nature medicine, 2018 [3]
- *Do Deep Generative Models Know What They Don't Know?*, ICLR, 2019 [8]

Papers available on my webpage (link)

- *Simple and scalable predictive uncertainty estimation using deep ensembles*, NeurIPS, 2017 [6]
- *Clinically applicable deep learning for diagnosis and referral in retinal disease*, Nature medicine, 2018 [3]
- *Do Deep Generative Models Know What They Don't Know?*, ICLR, 2019 [8]

Recent work on models combining $p(y|\mathbf{x})$ and $p(\mathbf{x})$

- *Hybrid models with deep and invertible features*, arXiv, 2018 [7]

Check out our ICML 2019 workshop
https://sites.google.com/corp/view/udlworkshop2019/

Thanks!

Acknowledgements:

- Alexander Pritzel and Charles Blundell
- Eric Nalisnick, Akihiro Matsukawa, Dilan Gorur and Yee Whye Teh

[1] C. M. Bishop. Novelty Detection and Neural Network Validation. 1994.

[2] L. Breiman. Random forests. *Machine learning*, 2001.

[3] J. De Fauw et al. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature medicine*, 2018.

[4] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[5] B. Lakshminarayanan. *Decision trees and forests: a probabilistic perspective*. PhD thesis, UCL, 2016.

[6] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017.

[7] E. Nalisnick, A. Matsukawa, Y. Teh, D. Gorur, and B. Lakshminarayanan. Hybrid models with deep and invertible features. 2018.

[8] E. Nalisnick, A. Matsukawa, Y. Teh, D. Gorur, and B. Lakshminarayanan. Do Deep Generative Models Know What They Don’t Know? In *ICLR*, 2019.

Backup slides

nature medicine

ARTICLES
https://doi.org/10.1038/s41591-018-0107-6

Clinically applicable deep learning for diagnosis and referral in retinal disease

Jeffrey De Fauw[1], Joseph R. Ledsam[1], Bernardino Romera-Paredes[1], Stanislav Nikolov[1], Nenad Tomasev[1], Sam Blackwell[1], Harry Askham[1], Xavier Glorot[1], Brendan O'Donoghue[1], Daniel Visentin[1], George van den Driessche[1], Balaji Lakshminarayanan[1], Clemens Meyer[1], Faith Mackinder[1], Simon Bouton[1], Kareem Ayoub[1], Reena Chopra[2], Dominic King[1], Alan Karthikesalingam[1], Cían O. Hughes[1,3], Rosalind Raine[3], Julian Hughes[2], Dawn A. Sim[2], Catherine Egan[2], Adnan Tufail[2], Hugh Montgomery[3], Demis Hassabis[1], Geraint Rees[3], Trevor Back[1], Peng T. Khaw[2], Mustafa Suleyman[1], Julien Cornebise[1,3,4], Pearse A. Keane[2,4*] and Olaf Ronneberger[1,4*]

The volume and complexity of diagnostic imaging is increasing at a pace faster than the availability of human expertise to interpret it. Artificial intelligence has shown great promise in classifying two-dimensional photographs of some common diseases and typically relies on databases of millions of annotated images. Until now, the challenge of reaching the performance of expert clinicians in a real-world clinical pathway with three-dimensional diagnostic scans has remained unsolved. Here, we apply a novel deep learning architecture to a clinically heterogeneous set of three-dimensional optical coherence tomography scans from patients referred to a major eye hospital. We demonstrate performance in making a referral recommendation that reaches or exceeds that of experts on a range of sight-threatening retinal diseases after training on only 14,884 scans. Moreover, we demonstrate that the tissue segmentations produced by our architecture act as a device-independent representation; referral accuracy is maintained when using tissue segmentations from a different type of device. Our work removes previous barriers to wider clinical use without prohibitive training data requirements across multiple pathologies in a real-world setting.

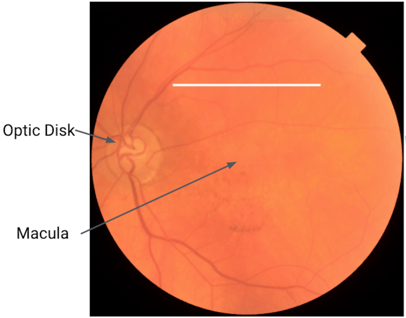
Medical imaging is expanding globally at an unprecedented rate[1,2], leading to an ever-expanding quantity of data that requires human expertise and judgement to interpret and triage. In many clinical specialities there is a relative shortage of this expertise to provide timely diagnosis and referral. For example, in ophthalmology, the widespread availability of optical coherence OCT has shown promise in resolving some of these criteria in isolation, but has not yet shown clinical applicability by resolving all three.

Results

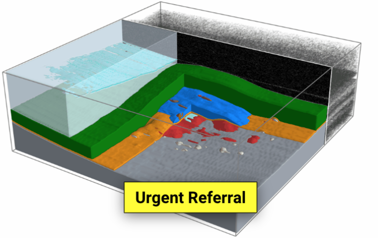
Clinical application and AI architecture. We developed our architecture in the challenging context of OCT imaging for oph-

Triage Recommendation for Patients with Eye Diseases using OCT scans

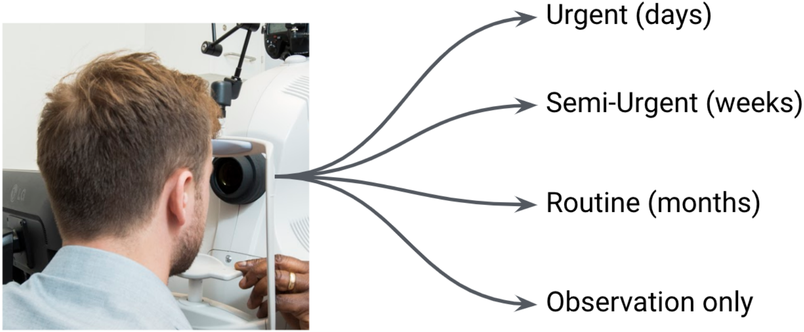
- Optical Coherence Tomography (OCT)
  - Creates a high-resolution 3D scan of the retina
  - OCT technique works like ultrasound but with light
- Collaboration with Moorfields Eye Hospital



Back of the eye (view through pupil)

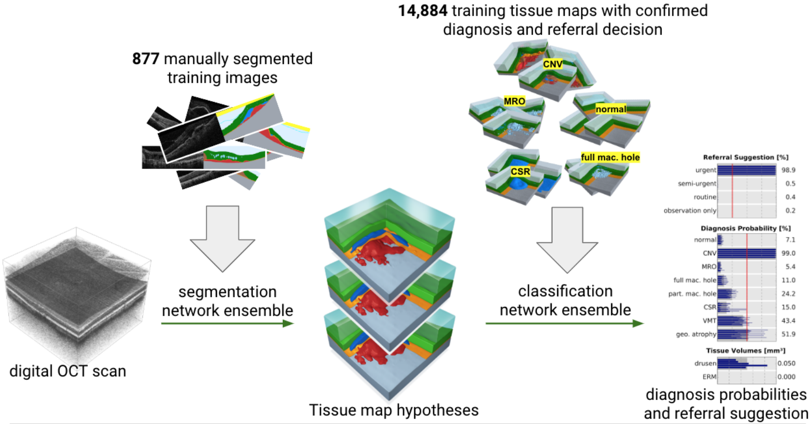


Use case: Referral suggestion from OCT scan



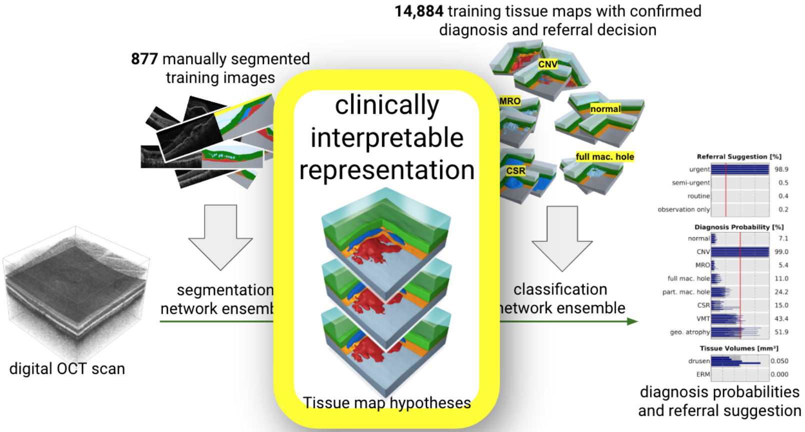
Two-Stage Architecture

- First: ensemble of segmentation networks to the OCT scan
- Second: ensemble of classification networks



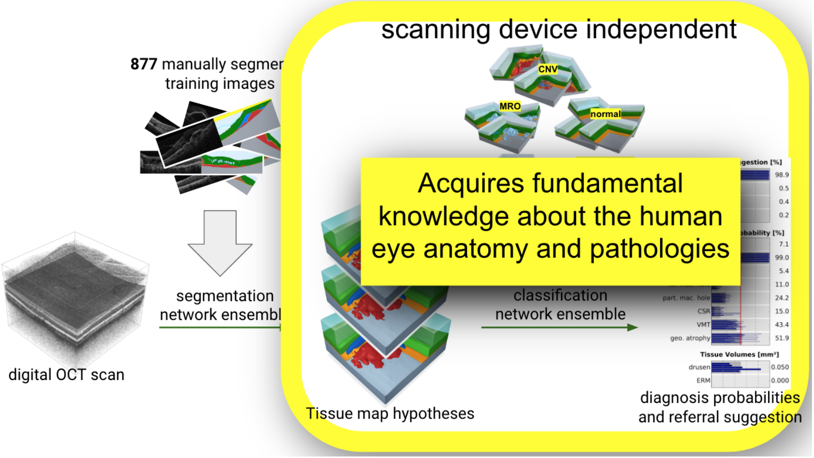
Two-Stage Architecture (continued)

- Segmentation map provides detailed, fully clinically interpretable representation.



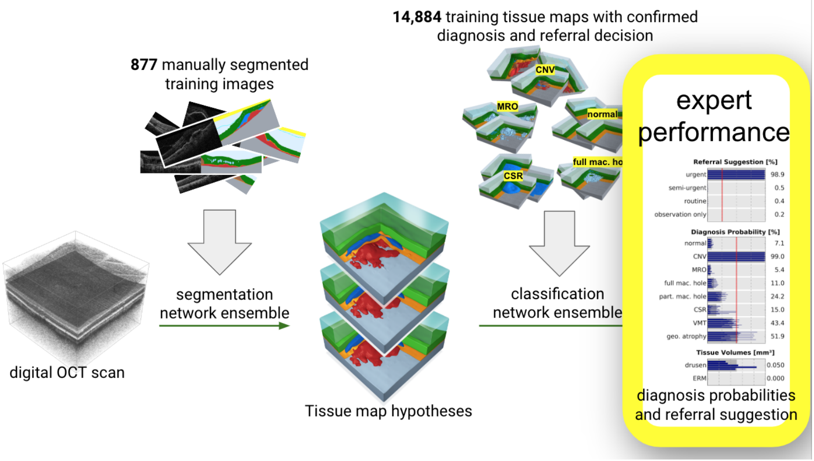
Two-Stage Architecture (continued)

- Second stage classification network learns knowledge that is independent of the used scanning device.

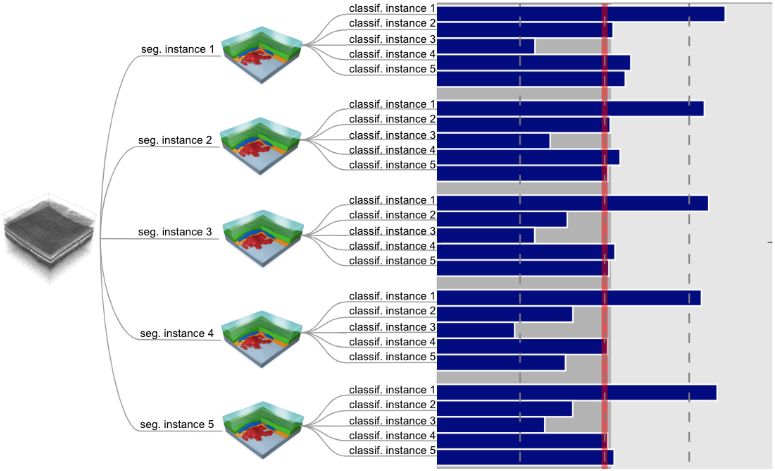


Two-Stage Architecture (continued)

- Our framework reaches the performance of human experts

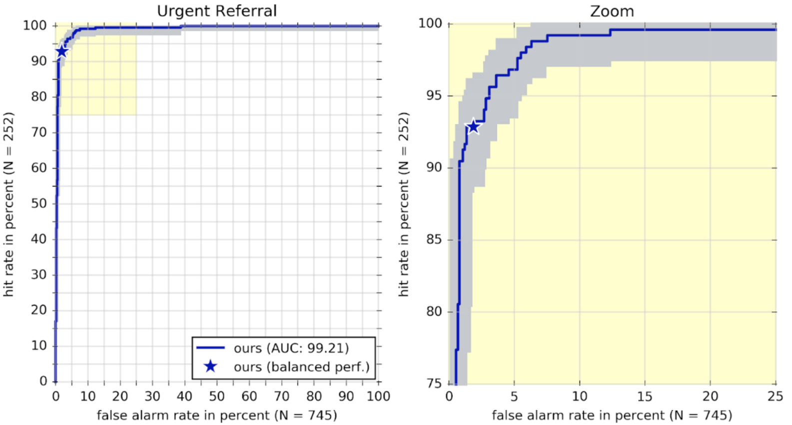


- Ensemble 5 segmentation instances and 5 classification instances to get 25 predictions for each diagnosis.



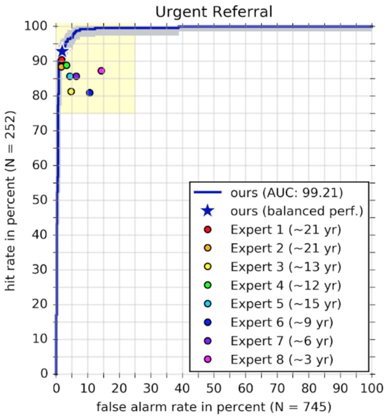
Receiver Operating Characteristic (ROC) Curve

- We achieve an area under the curve of 99.2



Receiver Operating Characteristic (ROC) Curve

- Evaluated human performance on this task using 8 experts
- Only two of the top experts from Moorfields with over 20 years experience were on par with our network



Full referral results

- Our method achieves similar results in the standard triage with 4 referral decisions too

Referral Decisions:

1. **Urgent** (within days)
2. **Semi-urgent** (within weeks)
3. **Routine** (within months)
4. **Observation only**

Our model
(OCT only)

Gold Standard Referral \ Predicted Referral	Urgent	Semi-urgent	Routine	Obser-vation
Urgent	234	5	13	0
Semi-urgent	3	225	2	0
Routine	10	2	250	4
Obser-vation	1	1	14	233

Expert 1
(OCT+fundus+notes)

Gold Standard Referral \ Predicted Referral	Urgent	Semi-urgent	Routine	Obser-vation
Urgent	228	4	20	0
Semi-urgent	3	223	4	0
Routine	2	7	254	3
Obser-vation	1	1	10	237

Expert 2
(OCT+fundus+notes)

Gold Standard Referral \ Predicted Referral	Urgent	Semi-urgent	Routine	Obser-vation
Urgent	231	8	13	0
Semi-urgent	1	226	3	0
Routine	11	1	250	4
Obser-vation	0	2	20	227