Probabilistic Model Ensembles for Predictive Uncertainty Estimation

Balaji Lakshminarayanan

balajiln@

DeepMind

Quantifying Uncertainty In Deep Learning

- Predict output distribution $p(y|x)$ rather than point estimate
  - Classification: output label y^* along with confidence
  - Regression: output mean and variance

Quantifying Uncertainty In Deep Learning

- Predict output distribution $p(y|x)$ rather than point estimate
  - Classification: output label y^* along with confidence
  - Regression: output mean and variance
- What's a "good" predictive uncertainty estimate?
  - Calibration
  - Higher uncertainty on out-of-distribution (OOD) examples

Quantifying Uncertainty In Deep Learning

- Predict output distribution $p(y|x)$ rather than point estimate
  - Classification: output label y^* along with confidence
  - Regression: output mean and variance
- What's a "good" predictive uncertainty estimate?
  - Calibration
  - Higher uncertainty on out-of-distribution (OOD) examples
- **Popular solution**: Bayesian deep learning (MCMC, VI)

Why 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

Why 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
- BMA satisfies the axioms of probability and protects against "Dutch books". **BMA is optimal if**:
  - "prior is correct" i.e. true model is within hypothesis class
  - true posterior can be computed exactly

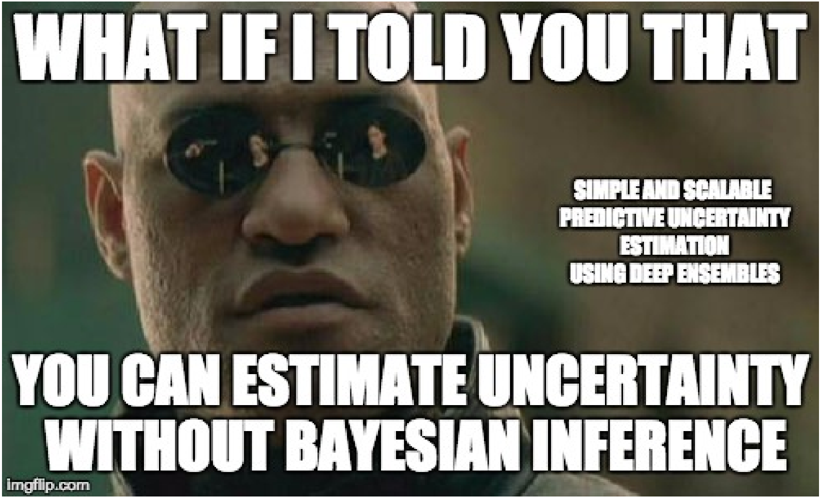
Why Bayesian deep learning?

- 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
- BMA satisfies the axioms of probability and protects against “Dutch books”. **BMA is optimal if**:
  - “prior is correct” i.e. true model is within hypothesis class
  - true posterior can be computed exactly

Is there an alternative to BMA for quantifying predictive uncertainty?

Yes!

Spotlight slide: BDL workshop @ NeurIPS 2016



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 $\Delta\mathbf{x} = -\epsilon \operatorname{sign}\left(\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(y|\mathbf{x})\right)$
   - 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

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 $p(y|\mathbf{x} + \Delta\mathbf{x})$
4. Train an **ensemble of M networks** with random initialization
5. Combine predictions at test time

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

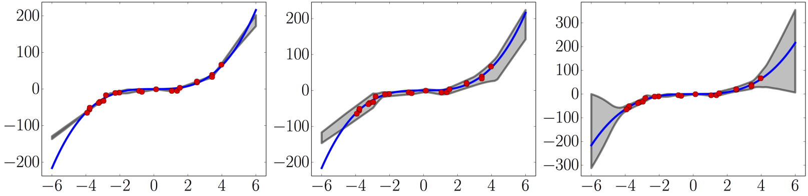
A Simple Recipe for Uncertainty Estimation

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

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

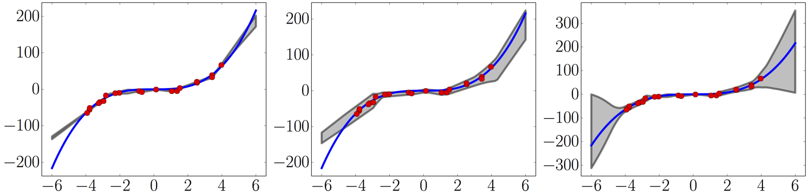
Model combination & not Bayesian Model Averaging

Results on a toy regression task



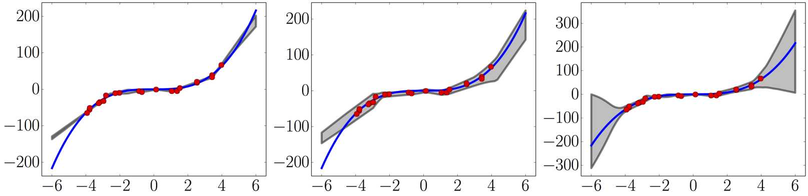
- Left plot: **non-probabilistic network**, use empirical variance between 5 networks as uncertainty

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**.

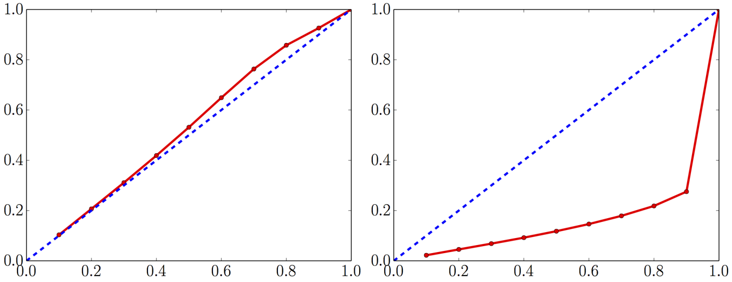
Results on UCI regression benchmark datasets

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

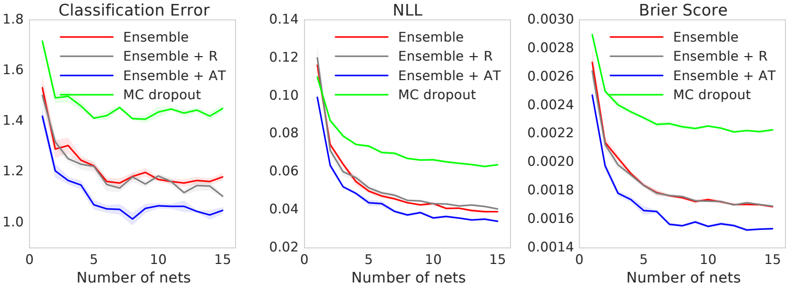
- Our method achieves better NLL, but slightly worse RMSE in some cases
- Even though non-Bayesian, our method is competitive with probabilistic backpropagation (PBP) and MC-Dropout

Calibration results on Year Prediction MSD

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

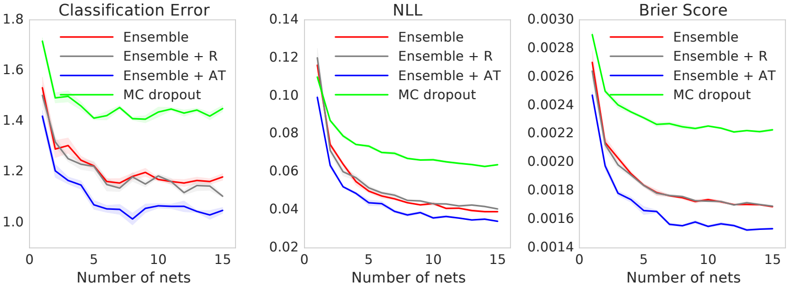


Classification Results on MNIST using MLP



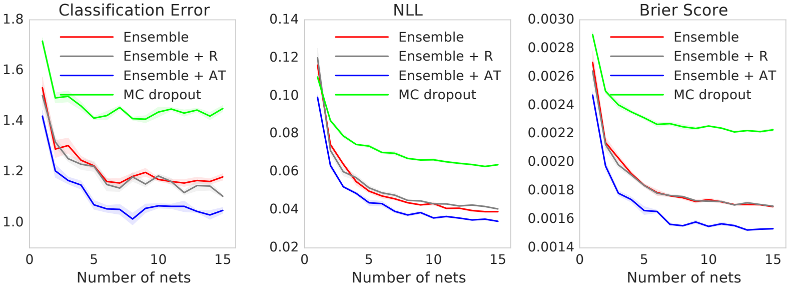
- **Ensembles lead to better predictive uncertainty**

Classification Results on MNIST using MLP



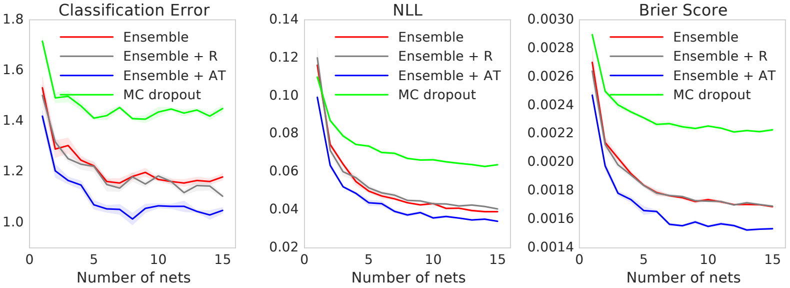
- **Ensembles lead to better predictive uncertainty**
- Adversarial training leads to further improvements

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

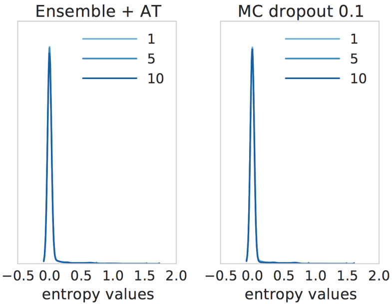
- Goal: check if the methods are more uncertain while testing on out-of-distribution (OOD) dataset.
- Setup:
  - Train on MNIST
  - Evaluate on known test set (MNIST) and unknown test set (NotMNIST) (both 28 x 28 gray-scale images)
- Also trained / tested on different datasets:
  - Train on SVHN / Test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

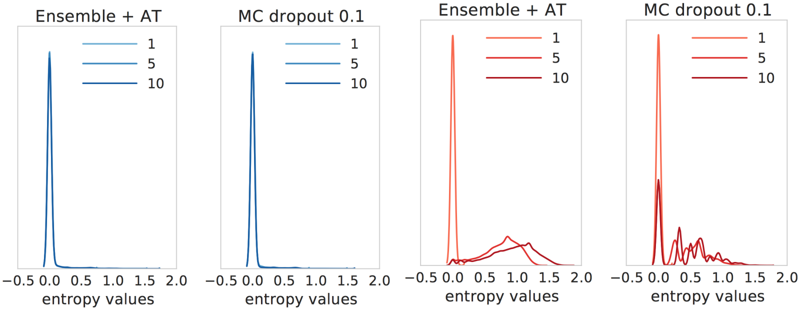
Predictive entropy on known & unknown inputs

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



Predictive entropy on known & unknown inputs

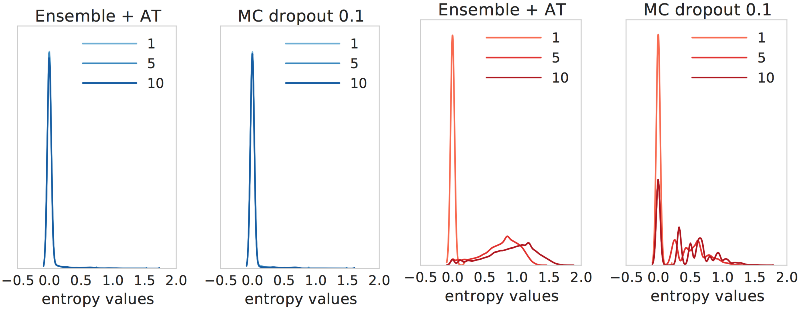
Train: MNIST. **Test**: MNIST + NotMNIST (out-of-distribution)



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

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

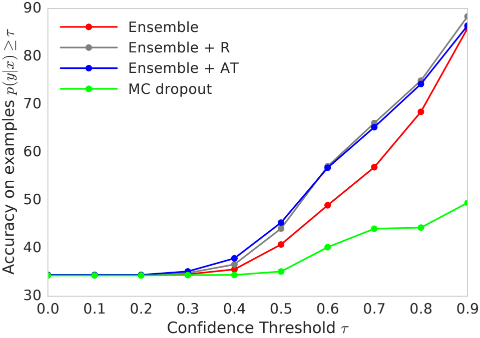
Model abstains from making prediction when confidence $< \tau$

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

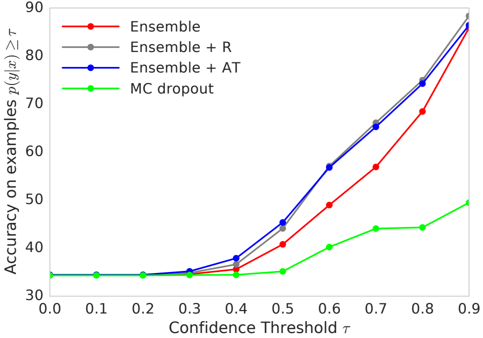
Model abstains from making prediction when confidence $< \tau$
Evaluate test accuracy only on examples where $\max_y p(y|\mathbf{x}) \geq \tau$
Train: MNIST. **Test**: MNIST + NotMNIST (out-of-distribution)



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

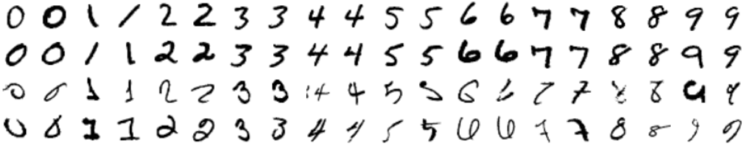
Accuracy Vs Confidence

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 [1] used bagging
  - Later work on Extremely Randomized Trees found bagging to be unnecessary if there was sufficient randomization [3]
  - (Non-Bayesian) Ensembles of probabilistic decision trees can give good uncertainty estimates in practice [4]

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.

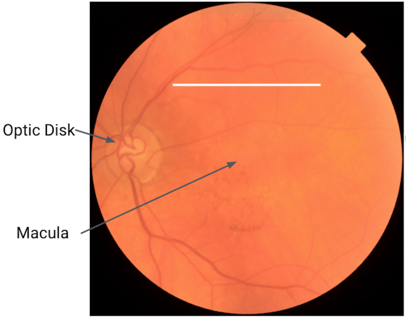
Medical imaging is expanding globally at an unprecedented rate[1,2], leading to an ever-expanding quantity of data that requires human expertise and judgement to interpret and triage. In many clinical specialities there is a relative shortage of this expertise to provide timely diagnosis and referral. 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

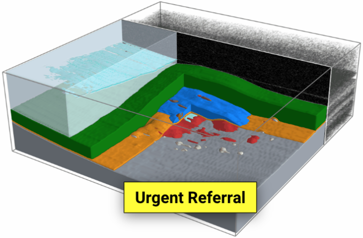
Clinical application and AI architecture. We developed our architecture in the challenging context of OCT imaging for oph-

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

Urgent (days)

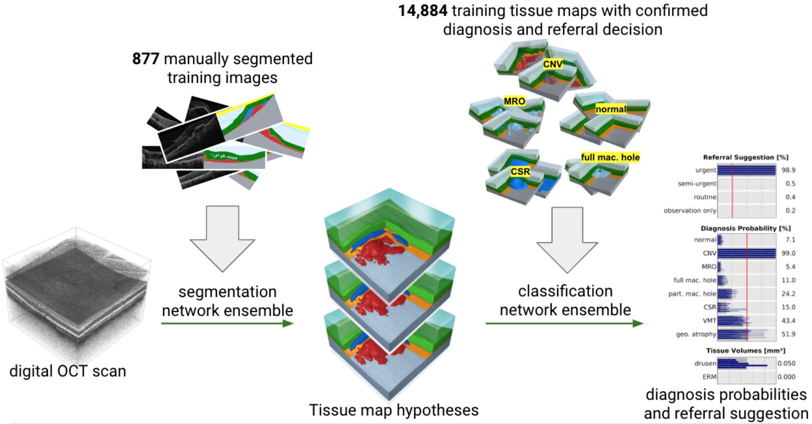
Semi-Urgent (weeks)

Routine (months)

Observation only

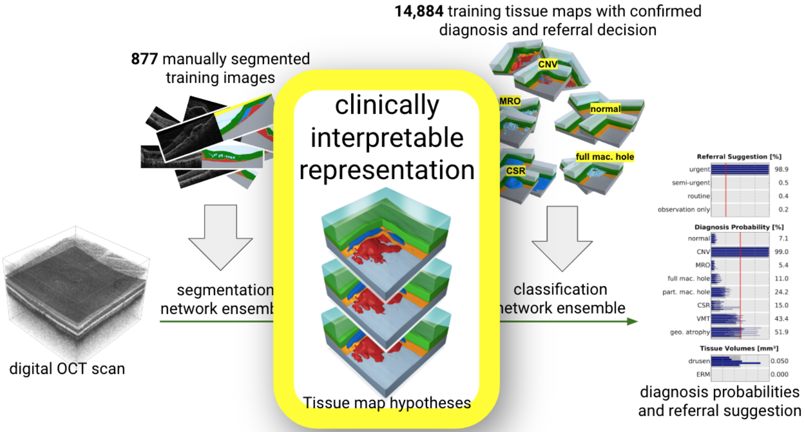
Two-Stage Architecture

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



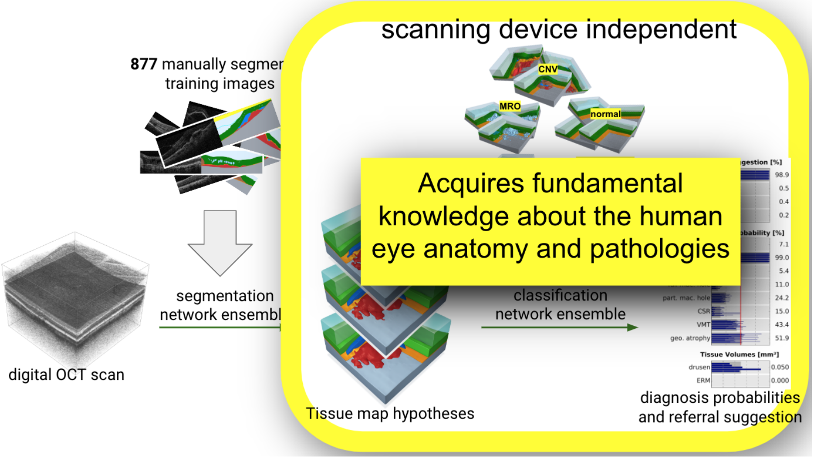
Two-Stage Architecture (continued)

- Segmentation map provides detailed, fully clinically interpretable representation.



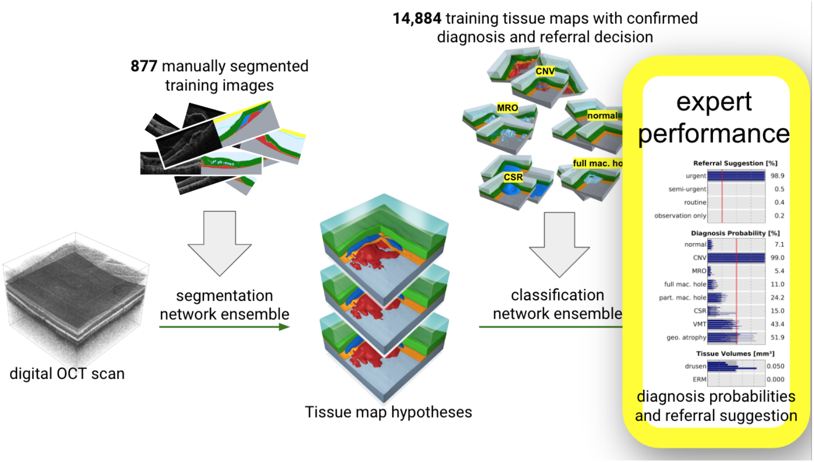
Two-Stage Architecture (continued)

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

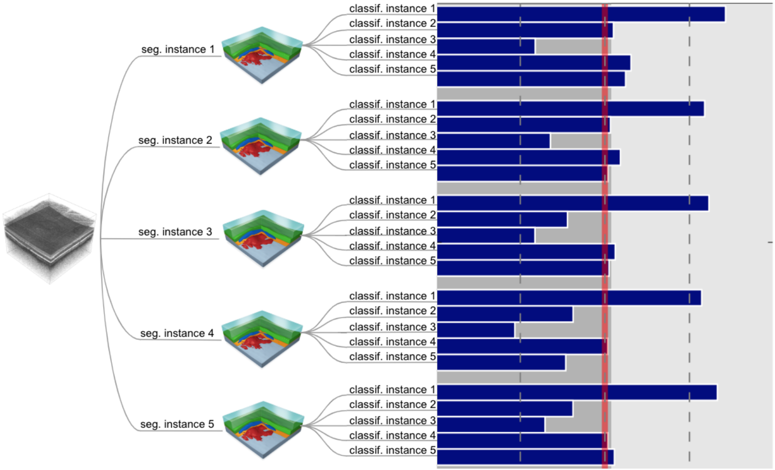


Two-Stage Architecture (continued)

- Our framework reaches the performance of human experts

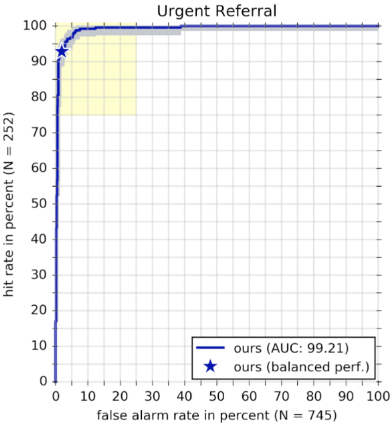


- Ensemble 5 segmentation instances and 5 classification instances to get 25 predictions for each diagnosis.



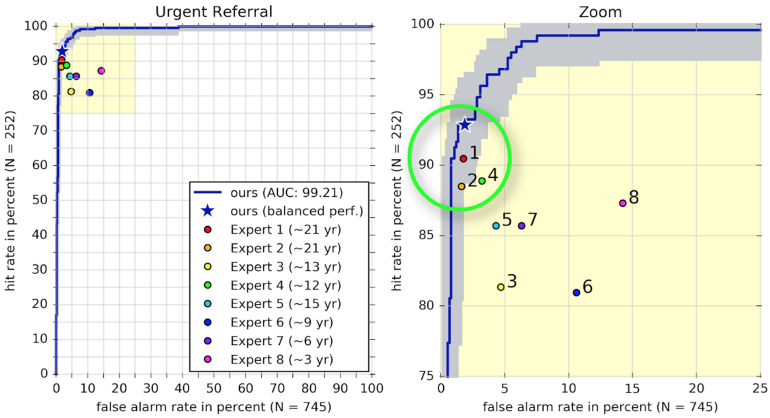
Receiver Operating Characteristic (ROC) Curve

- 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

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

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

Papers available on my webpage (link)

- *Simple and scalable predictive uncertainty estimation using deep ensembles*, NeurIPS, 2017 [5]
- *Clinically applicable deep learning for diagnosis and referral in retinal disease*, Nature medicine, 2018 [2]

Thanks!

Acknowledgements:

- Alexander Pritzel and Charles Blundell
- Olaf Ronneberger and other co-authors

[1] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] J. De Fauw et al. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature medicine*, 24(9):1342, 2018.

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

[4] B. Lakshminarayanan. *Decision trees and forests: a probabilistic perspective*. PhD thesis, UCL (University College London), 2016.

[5] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.