PAC-Bayes Inspired NAS Without Training

James Bowden, Steven Csaposs, Thomas Hoffmann

How can we quickly design neural networks in an automated, theoretically-motivated way?

- It is unlikely that a single architecture will outperform all others on every task. Neural Architecture Search (NAS) generally provides a strong architecture for a given task.
- However, NAS tends to be very slow and resource intensive (hours on low end, days on high end)
- We investigate PAC-Bayes evidence motivated objectives to perform NAS without training, with the goal to improve NAS efficiency

Relevant Literature

- Neural Architecture Search without Training by Mellor et al. introduces a faster NAS without training by minimizing the correlations in the network
- Liu et al. in Are Labels Necessary for Neural Architecture Search? find that unsupervised objectives can perform similarly to supervised objectives in NAS
- Previous works did not consider unsupervised objectives with no training nor using many initializations

Methods

High level:

- NAS via Bayesian optimization over MNIST dataset
- Using CNN architectures (less trivial than feedforward)
- Plug-in objective functions
- Use few (100) (un)labeled images from training set

Computing evidence for feedforward NN (lecture):

1. Sharpness/flatness of the weight space
2. Breaking symmetries
3. Infinite width proxy

Generally, making modeling assumptions to derive principled evidence metrics.
As such, (1) and (2) are nontrivial to compute, and (3) doesn't make much sense in the context of CNNs.

Objective metrics motivated by PAC-Bayes evidence:

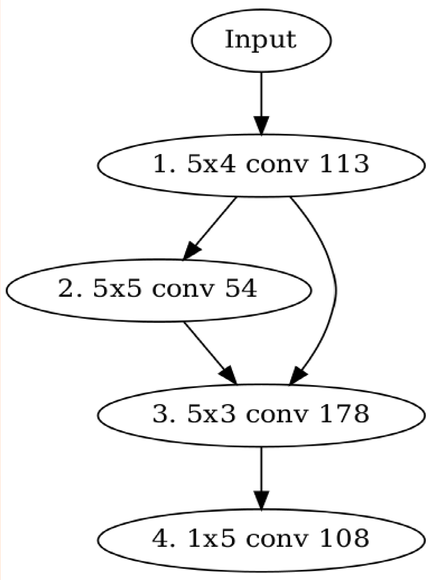
$$e = \text{Vol(solutions)} / \text{Vol(weight space)}$$

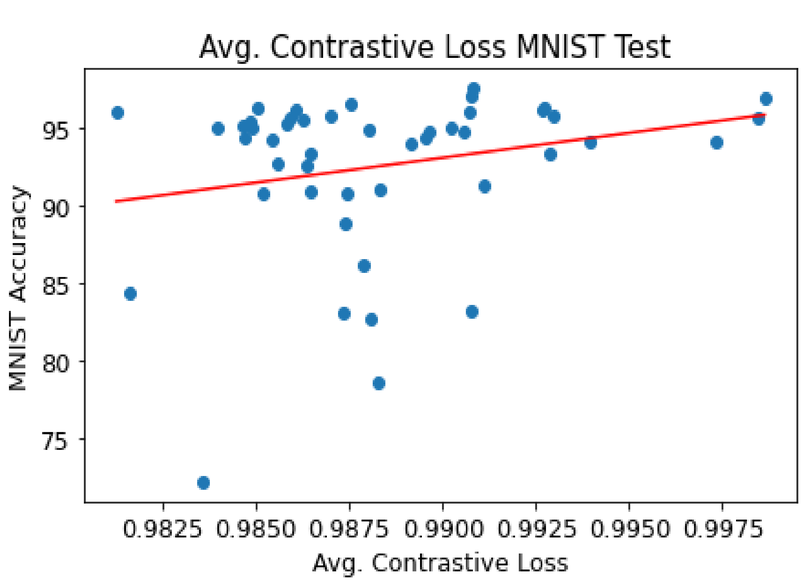
We expect random solutions from an architecture with more evidence to generalize better. This motivates our general method:

1. Generate architecture via Bayesian Optimization
2. Randomly initialize network weights (untrained)
3. Compute metric
4. Repeat 2-3 n times (e.g. 100)
5. Aggregate and return metric

We plug in hand-designed objective functions and report their predictive behavior w.r.t. test error of trained models.

Contrastive Objective

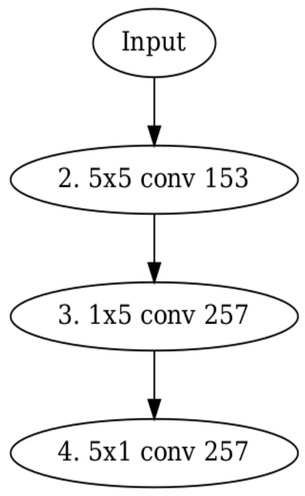


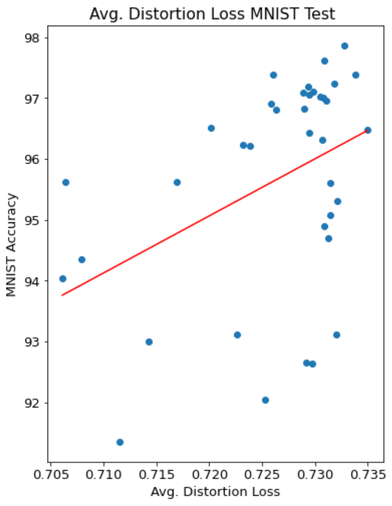


- Use cosine distance triplet loss with two distinct augmentations of the image to create anchor and positive sample, select a negative sample randomly
- Average over 100 weight initializations
- Search 500 architectures, 100 unlabeled MNIST images
- 0.056 R^2
- Only 2 architectures beat highest-scoring architecture

Perturbations via MC Dropout

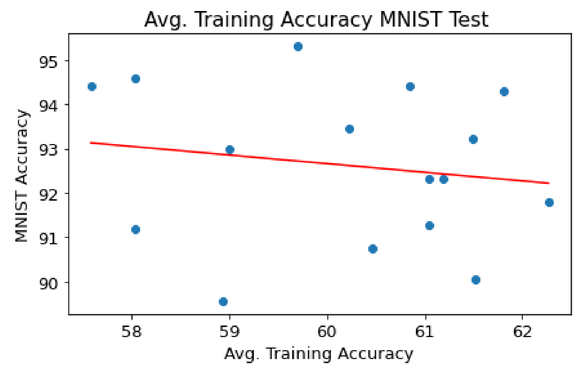
- Perturb weights, evaluate "alpha-robustness" as weight space flatness
- 30 weights, apply MC dropout perturbation x5
- Measure avg. cosine distance of predictions
- "Bayesian sampling"
- 0.17 R^2





Naive Training Accuracy

- Naive metric using average accuracy over 600 training images
- 100 untrained weight init.
- Performs pretty poorly; slight negative correlation



Conclusions

- Each evaluation of our objectives is over 50x faster than training.
- The average label-free cosine distance triplet loss over 100 random initializations of the network is predictive of the trained accuracy of the neural network and is a NAS objective.
- Measuring robustness to perturbation in the weight space via MC Dropout is also predictive of test accuracy and has potential as a NAS objective. Would like further investigation of perturbation approaches, including adversarial robustness.