LEARNING THE INTRINSIC DIMENSION OF COMPLEX LARGE DATASETS

By

Joseph M. Weaver

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Statistics - Master of Science

2024

ABSTRACT

Bayesian methods are used to estimate intrinsic dimensionality (ID) and to perform dimensionality reduction for a large complex dataset. Using 130,000 images over 100 categories, we developed a process to reduce the dimensionality to a very small size while preserving the ability to classify the images. The novelty of our approach is two-fold, 1) 2NN estimation of target dimensionality is now used as a prior distribution, and 2) the overall mapping is generated by a Bayesian neural network which then selects an appropriate intrinsic dimension based on the prior, variational inference, and multidimensional scaling.

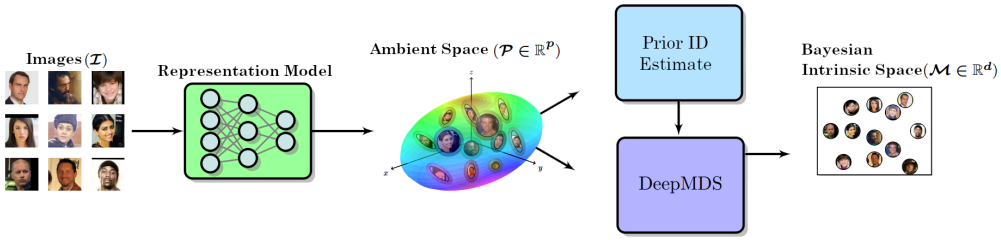
TABLE OF CONTENTS

1 Introduction

Intrinsic dimensionality (ID), as coined by [Bennett, R. S. 1965], defines a lower bound of the number of free parameters capable of producing a close approximation of an originating signal or data. Instead of preserving all information, we discard superfluous information preserving only what is needed classification. As such, we define intrinsic dimensionality as the number of parameters d of the lowest d-dimensional space such that some function of a p-dimensional space is mostly preserved, such that $d << p$.

Finding such a manifold is motivated by exponentially increasing data collection and the need to be able to categorize it. Our attention is directed towards image data, where images are quantified in megapixels, also referred to statistically as millions of parameters. This leads to a classical large-p small-n problem, which is computationally costly and leads to model complexity. Naturally, this drives the need to reduce the p-dimensions to a much smaller d-dimensional manifold.

Figure 1: Mapping of Raw Images to Intrinsic Manifold



For this reason, many techniques have been proposed to reduce dimensionality. Our project focuses on the branch of dimensionality reduction dealing with nonlinear projections to a lower dimensional manifold, similar to multidimensional scaling. It closely follows two works; 1) On the Intrinsic Dimensionality of Image Representation [Gong, Boddeti, and Jain 2019], and 2) Variational Bayes Ensemble Learning Neural Networks With Compressed Feature Space [Liu, Bhattacharya, and Maiti 2021].

[Gong, Boddeti, and Jain 2019] proposed a dimensionality reduction process that first estimates the intrinsic dimension using the K-nearest neighbor approach and the Intrinsic Dimensionality Reduction Algorithm (IDEA). Second, they create a map to this intrinsic dimension using a deep neural network, they called DeepMDS. Their process was able to maintain a substantially higher degree of accuracy over PCA, Isomap, and denoising autoencoders when reducing to very small dimensions.

The important result from [Gong, Boddeti, and Jain 2019] demonstrates a reduction of ImageNet/ResNet34 512-dimension image representation down to an intrinsic dimension of 19, with a loss of accuracy of 6%, (39% down to 33%). As our first objective, we will recreate

this experiment as a baseline.

In [Liu, Bhattacharya, and Maiti 2021], they propose a variational inference-based approach with Bayesian model averaging in the context of a Bayesian Neural Network to improve upon the prior paper. This showed remarkable improvements in the case of synthetic datasets such as Hyperspheres and Swissrolls, and a real dataset of handwritten digits, MNIST. Given this success, our second objective will test this process on the larger dataset, namely the ImageNet/ResNet34 512-dimension image representation and compare our results to the baseline described above.

Finally, we propose adding the 2NN estimation technique described in [Gong, Boddeti, and Jain 2019] as a prior distribution to the Bayesian neural network described in [Liu, Bhattacharya, and Maiti 2021]. Thus, as third objective we seek to obtain intrinsic dimensionality estimate of the ImageNet data with a quantified uncertainty.

Altogether, this study 1) recreates the experiment described in [Gong, Boddeti, and Jain 2019], 2) demonstrate the effectiveness of BNN in dimensionality reduction at scale, and 3) quantify the uncertainty of the intrinsic dimension using a 2NN prior.

2 Method/Approach

To implement our novel approach, we first estimate the intrinsic dimension using our two nearest neighbor method, then proceed to train our map from ambient space to intrinsic space using the TwoNN information as prior to quantify the uncertainty of each intrinsic dimension.

2.1 Bayesian Prior for ID Estimation

To estimate our intrinsic dimension, we use a simple Bayesian formulation

$$p(d|\mu) \propto p(\mu|d)p(d)$$

where $\mu \in R^n$ is the ratios of two nearest neighbors, and d is our intrinsic dimension. $p(\mu|d)$ is the likelihood of the $Pareto(1, d)$ distribution.

$$\prod_{i=1}^{n} \frac{d}{\mu_i^{d+1}} = d^n e^{-(d+1)v}$$

where $v = \sum_{i=1}^{n} ln(\mu_i)$. $p(d) \sim Gamma(a, b)$ is our minimally informative conjugate prior.

$$p(\mu|d)p(d) = d^n e^{-(d+1)v} \times \frac{b^a}{\Gamma(a)} d^{a-1} e^{-bd} = c d^{n+a-1} e^{-bd-(d+1)v}$$

where $c = \frac{b^a}{\Gamma(a)} e^{-v}$, which implies that $p(d|\mu) \sim Gamma(a + n, b + v)$.

2.1.1 Two Nearest Neighbor estimation of ID

Consider data $X \in R^{n \times p}$, and let X_i denote the i^{th} row vector. Let $D \in R^{n \times n}$ be the distance matrix such that element $d_{ij} = \|X_i - X_j\|_2$ is the Euclidean distance between the i^{th} and j^{th} rows of data X. Let $r_{i,1}, r_{i,2}$ be the two smallest distances (e.g. Two nearest neighbors) in the i^{th} row of D excluding $d_{ii} = 0$. Let $\mu_i = r_{i,2}/r_{i,1}$ be the ratio of the two nearest neighbors distance for i^{th} row of X. Then $\mu_i \sim Pareto(1, d)$ which the unbiased MLE for d, $\hat{d} = \frac{n-1}{\sum_{i=1}^{n} \mu_i}$ [Allegra, et. al. 2020].

2.1.2 Derivation of Pareto shape parameter d MLE

Due to conflicting material for the unbiased estimator for Pareto shape parameter, we verify the correct estimator. Consider minimization with respect to d of the log-likelihood of a $\mu_i \sim Pareto(1, d)$.

$$\min_d \ell(d, \mu) = \min_d ln \prod_{i=1}^{n} \frac{d}{x\mu_i^{d+1}} = \min_d nln(d) - (d+1) \sum_{i=1}^{n} ln(\mu_i) \frac{\partial \ell}{\partial d} = \frac{n}{d} - \sum_{i=1}^{n} ln(\mu_i) = 0$$

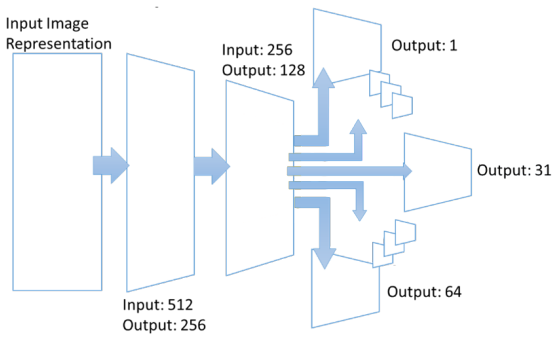
According to [Rytgaard 1990], $E(\hat{d}) = \frac{n}{(1-n)}d$. We correct for the bias and verify what is described in [Denti, et. al. 2022]. Thus the MLE of d is,

$$\hat{d} = \frac{n-1}{\sum_{i=1}^{n} ln(\mu_i)}$$

2.2 Ensemble

To create a map from high dimensional space, we describe a sequence of 3 neural networks with decreasing input and output dimensions, starting with an input dimension of 512 and reducing down to 256 and 128. At the last stage, we create a NN that reduces from 128 to every dimension between 1 and 64, as diagrammed in Figure 2.

Figure 2: Hierarchy of models



2.3 Bayesian Neural Network

Largely based on [Gong, Boddeti, and Jain 2019], [Liu, Bhattacharya, and Maiti 2021], and [Bishop 2006], the following is a mathematical formulation of the Bayesian neural network we use in this study.

Given data $\mathcal{D} = [X|Y]$ where the image category $Y \in (1, 2, ..., k)^n$ and image representation $X \in R^{p \times n}$ where row vector X_i is assumed to be generated from Poisson point process. Let $l \in (0, 1, ..., L)$ represent the index of each layer of the network, where $l = 0$ is the input layer, and $l = L$ is the output layer. Let,

- $p^{(l)} \in N$ be the dimension of layer l,
- $a^{(l)} \in R^{p^{(l)}}$ be the output values of $p^{(l)}$-dimensional vector,
- $W^{(l)} \in R^{p^{(l-1)} \times p^{(l)}} \sim N(\mu_W, log(1 + e^{\rho_W}))$ be the Bayesian weight matrix for layer $l = 1, 2, ..., L$,
  - $\mu_w \in R^{p^{(l-1)} \times p^{(l)}}$ be the parameters representing the mean of the weight,

  - $\rho_w \in R^{p^{(l-1)} \times p^{(l)}}$ be a parameter for an unconstrained transformed variance such that $\sigma^2 = log(1 + e^{\rho})$,

- $b^{(l)} \in R^{p^{(l)}} \sim N(\mu_V, log(1 + e^{\rho_V}))$ be the Bayesian bias vector of layer $l = 1, 2, ..., L$,
  - $\mu_v \in R^{p^{(l)}}$ be the mean of the bias,
  - $\rho_v \in R^{p^{(l)}}$ be the transformed variance of the bias,
- $f^{(l)}_{LReLU}(.) : R^{p^{(l)} \to R^{p^{(l)}}}$ be the LeakyReLU activation vector function of layer $l = 1, 2, ..., L - 1$.

Then the general form of our model is

$$a^{(0)} = X,$$

$$a^{(l)} = f^{(l)}_{LReLU}(W^{(l)} a^{(l-1)} + b^{(l)}), l = 1, 2, ..., L - 1,$$
$$Z = W^{(L)} a^{(L-1)} + b^{(L)},$$

where Z is the estimated reduced dimensionality vectors. In a Bayesian neural network, parameters are represented by distributions and during the feed forward process the weight is approximated by $W^{(l)} = \mu_w + zlog(1 + e^{\rho_w})$ where $z \sim MVN(0, I)$ is randomly generated. The bias vector $b^{(l)} = \mu_v + zlog(1 + e^{\rho_v})$ is evaluated similarly.

To complete the description of the BNN, we define two different types of intermediate layers, Linear and BatchNorm. The layers described above are actually a combination of a Linear layer, followed by a BatchNorm layer which is then fed into our activation function. Both Linear and BatchNorm layers have Bayesian parameters, but the later also aggregates the inputs defining $\mu = E(a^{(l-1)})$ and $\Sigma = Var(a^{(l-1)})$ and normalizes them as follows,

$$a^{(l)} = W^{(l)} \frac{a^{(l-1)} - \mu}{\sqrt{\Sigma}} + b^{(l)}.$$

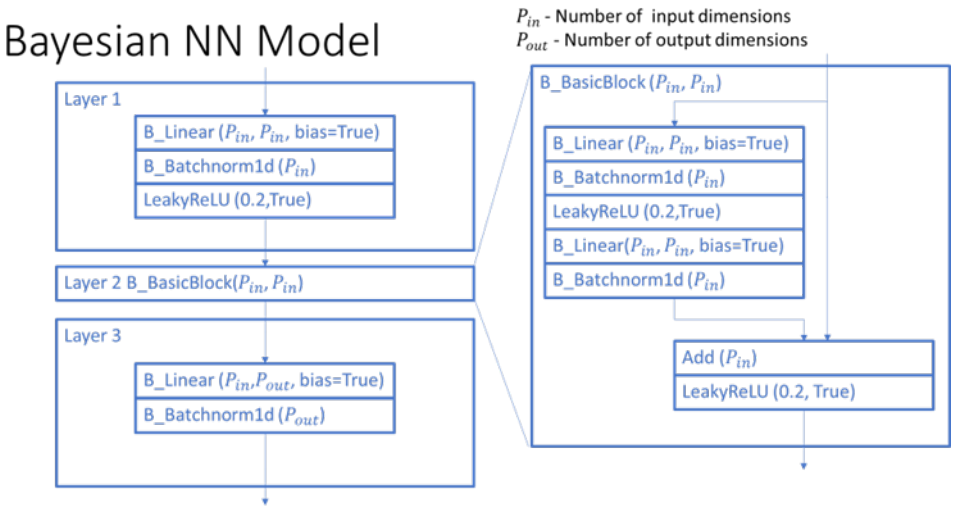
Finally, we implement a skip layer which forwards the output of our first 2 layers to our last 2 layers to enhance the training of our input layers. Altogether this results in a network as diagrammed in Figure 3.

2.3.1 Loss Function

In training our neural network, the loss function uses the estimated lower bound (ELBO) ([Liu, Bhattacharya, and Maiti 2021]). This function is,

$$Loss = KL + LL$$

Figure 3: Bayesian NN Architecture



$$KL = \sum_{l=1}^{L}\left[\sum_{i=1}^{p^{(l)}}\sum_{j=1}^{p^{(l-1)}} KL(W_{ij}^{(l)}||\phi) + \sum_{i=1}^{p^{(l)}} KL(b_i^{(l)}||\phi)\right],$$

where $W_{ij}^{(l)}$ is the ij^{th} weight random parameter on layer l, $b_i^{(l)}$ is the i^{th} bias random parameter on layer l, and $\phi \sim N(0, \sigma_0^2)$.

$$KL(P||Q) = ln(\frac{\sigma_Q}{\sigma_P}) + \frac{\sigma_P^2 + (\mu_P - \mu_Q)^2}{2(\sigma_Q^2)} - \frac{1}{2}$$

which is the reduce form of KL-divergence for normal densities.

$$LL = \sum_{i=1}^{n}\sum_{j=1}^{n}\Big[\|X_i - X_j\|_2 - \|Z_i - Z_j\|_2\Big]^2$$

where X_i is the i^{th} input image representation, and Z_i is the associated reduced image representation.

2.4 Optimization

Our model training follows the classical feed forward - backwards propagation techniques of neural network utilizing the AdamW optimizer. The process first initializes the parameters of our Bayesian weights and biases. Then in successive steps we feed forward a batch of input image representations resulting in lower dimensional image representations. We then calculate our loss gradient ∇f and proceed to back-propagate, updating the parameters of our Bayesian weights and biases.

2.4.1 Initialization

For each element $W_{ij}^{(l)}$ for $l = 1, 2, ..., L$, let $\mu_{i,j}^{(l)}$ be randomly generated from $uniform(-.6, .6)$, and $\Sigma_i^{(l)} = log(1 + e^{-6})$ and likewise for each element $b_i^{(l)}$ for $l = 1, 2, ..., L$.

2.4.2 AdamW

AdamW optimization [Loshchilov and Hutter 2017] is a variant of gradient descent that incorporates adaptive learning rates, momentum, and weight decay that is specially designed for training deep neural networks. Let $\beta_1 = 0.9$, and $\beta_2 = 0.999$ be how much the first two moments are maintained between steps. Let $\lambda = 1e-4$ be the weight decay, $\gamma_0 = 3e-4$ be the initial learning rate, $\epsilon = 1e-8$ be a small adjustment to prevent division by zero, and $m_0 = 0, v_0 = 0$ be the initial value of the first two moments.

For t in $1, 2, ...$ apply the following update rule until there is not a significant change in loss,

$$\theta_t = \theta_{t-1} - \gamma_t \lambda \theta_{t-1} - \gamma_t \frac{m_t}{\sqrt{v_t} + \epsilon}$$

where the term $-\gamma_t \lambda \theta_{t-1}$ implements the weight decay, and momentum is formulated as the following,

$$m_t = \frac{\beta_1 m_{t-1} + (1 - \beta_1)\nabla f(\theta_{(t-1)})}{1 - \beta_1^t}, v_t = \frac{\beta_1 v_{t-1} + (1 - \beta_2)\nabla f(\theta_{t-1})^2}{1 - \beta_2^t},$$

Finally, the variable learning rate, Cosine Annealing, is implemented as follows,

$$\gamma_t = \gamma_{min} + \frac{1}{2}(\gamma_{max} - \gamma_{min})(1 + cos(t/t_{max}))$$

2.5 Accuracy

The following section details the algorithms we use to determine if the projection has maintained the same ability to classify as the input data. Primarily, we use mAP as both a baseline and post training metric to quantify the accuracy or change in accuracy as a result of data reduction. The goal being to maintain a nearly identical mAP value between stages of reduction to a minimum intrinsic dimension.

2.5.1 mAP

The mean average precision (mAP) is a standard averaging of the average precision for each category. For instance, the ImageNet data contained 100 different categories and we can calculate the average precision for each category. Thus the mAP would be;

$$mAP = \frac{1}{100} \sum_{i=1}^{100} AP_i$$

Where the AP_i is the average precision for the i^{th} category.

2.5.2 Average Precision

Average precision is a measure of Type I and Type II errors that take into account the trade-off between the two values for different selections of a threshold. Specifically, average precision is the area under the precision-recall curve and is in practice estimated by summing the precision $P(r)$ times the change in recall $\Delta r(k)$ over all k changes in the recall.

$$AP = \sum_{i=1}^{n} [P(k)\Delta r(k)]$$

2.5.3 Precision and Recall

Precision and Recall are measures of Type I and Type II errors, respectively. Precision is the ratio of true positives over all the positive predictions $TP/(TP + FP)$. Thus when the precision is 1.0, this indicates that there are no false positives (Type I errors). Recall is the ratio of true positives over actual positives $TP/(TP + FN)$. Thus when recall is 1.0 when there are no false negatives (Type II errors). Therefore, as a threshold for deciding if a prediction is positive or negative increases the precision decreases and recall increases, thus creating a trade-off between the two. In a perfect prediction scenario, there would not be any Type I or II errors, and therefore precision and recall are always 1.0, and therefore the maximum value of average precision is also 1.0.

3 Theory

To derive our loss function we combine concepts from multidimensional scaling (MDS) with variational inference (VI).

3.1 Multidimensional Scaling

At a high level, MDS is a technique that preserves the pairwise similarity or dissimilarity between a set of points that have been projected to lower-dimensional space.

In our model, the mapping between ambient space and intrinsic space is performed by a neural network $f_\theta(.)$ with weights and biases of θ such that $Z_i = f_\theta(X_i)$ where X_i and Z_i are the ambient and reduced vector representations respectively.

To preserved the pairwise difference in Euclidean distance we minimize $d_{ij} = \Big[\|X_i - X_j\|_2 - \|Z_i - Z_j\|_2\Big]^2$ which are assumed to be normally distributed $N(0, \sigma^2)$, thus resulting in the following likelihood,

$$\log L(\theta) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\Big[\|X_i - X_j\|_2 - \|Z_i - Z_j\|_2\Big]^2 + \frac{1}{2}\log(2\pi\sigma^2)$$

Of which the scalar $\frac{1}{2\sigma^2}$ and constant $\frac{1}{2}\log(2\pi\sigma^2)$ can be ignored during optimization simplifying to,

$$\log L(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}$$

3.2 Variational Inference

Utilizing a Bayesian neural network composed of a myriad weights and biases represented by distributions with unspecified parameters θ, we employ VI to train these distribution parameters.

We define our posterior distribution

$$p(\theta|data) \propto e^{\log L(\theta) + log(p(\theta))}$$

where $p(\theta) \sim N(0, \sigma_0^2 I)$ is a Gaussian prior and $\log L(\theta)$ is the MDS loss described above. Then let $q(\theta)$ be a variational family such that $q(\theta) \sim N(\mu, \Sigma)$ where μ and Σ are unknown parameters in θ.

To find the variational posterior, we choose $q(\theta)$ such that the Kullback-Leibler divergence $KL(q(\theta), p(\theta|data))$ is minimized. Or equivalently minimize,

$$E_q(\log q(\theta) - \log p(\theta|data)),$$

$$= E_q(\log q(\theta) - \log L(\theta) - \log p(\theta)) + C,$$

where C the proportionality constant in $p(\theta|data)$ which is ignored under optimization. So finally we minimize the following,

$$-E_q(\log L(\theta)) + KL(q, p)$$

4 Numerical Studies

4.1 Dataset (ImageNet100)

We obtained a cluster-sampled of 100 subject categories containing 1350 images for each subject from the sizable ImageNet image repository used in the [Gong, Boddeti, and Jain 2019]'s paper. The images were processed using ResNet34, creating 512-dimensional image representations. These image representations encapsulate quantified visual information that differentiates images among various categories. The data was separated into training data by performing a stratified sample without replacement of 1300 image vectors from each the 100 categories, and the remaining 50 vectors became the test data. The training data is use to calculate our TwoNN prior ID Estimation and also use to train our BNN for mapping to lower dimension. The test data is used to validate the accuracy of the model by calculating the mAP value of the original data and reduced data.

4.2 TwoNN

In this study, we estimate the intrinsic dimensionality of a high dimensional dataset using TWO-NN and the MLE of the Pareto shape parameter. Specifically, we calculate the Euclidean norm between all image representations and find the two nearest neighbors to each point. We then calculate the ratio of two nearest neighbors and estimate the shape parameter of the Pareto distribution, then find the gamma posterior distribution using the Pareto and its conjugate prior.

Figure 4: Estimated ID based on sample size

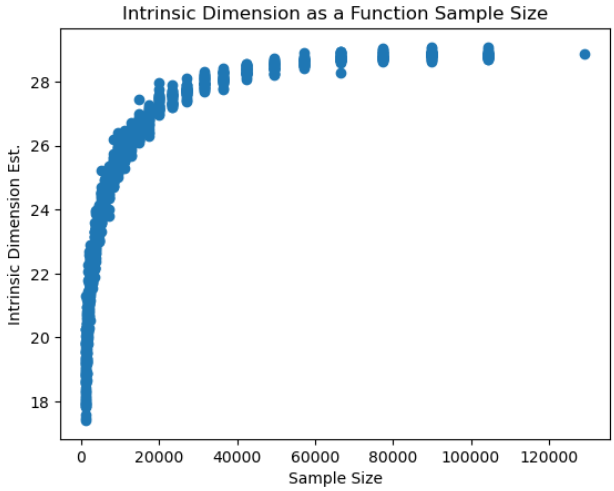
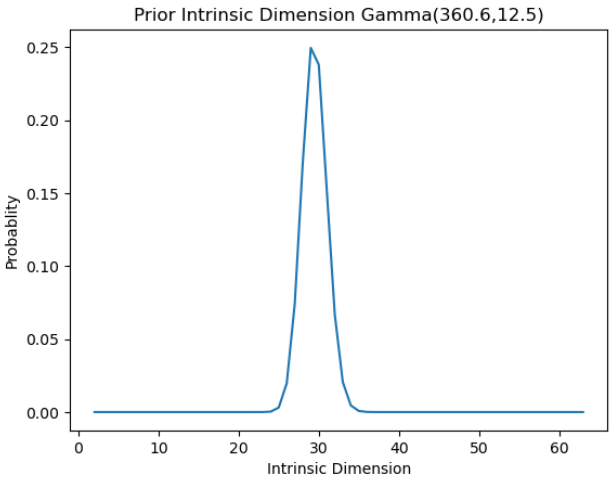


Figure 5: ID estimate distribution



To verify that the ID calculation is consistent, we calculated the TwoNN for various stratified sample sizes increasing up to our entire ImageNet100 sample. We collect 30 estimates per sample size. As you can see in Figure 4, the ID estimate increases as sample size

increases, but only up to a certain sample size.

As part of this experiment, we had to overcome the challenge of calculating and storing a distance matrix with 17 billion elements and a memory requirement of 540 GB. Since the ratio calculation for a single point was independent of other calculations, we have implemented parallelization techniques to distribute our computation across a network of 130,000 concurrent processes.

Then given the posterior distribution described above, we estimate the intrinsic dimension to be a $Gamma(359.5831, 11.4508)$ with a mean of 28.96 and variance of 2.33. As graphed in Figure 5 this distribution that will become the prior for our final ID estimation as a result of our neural network.

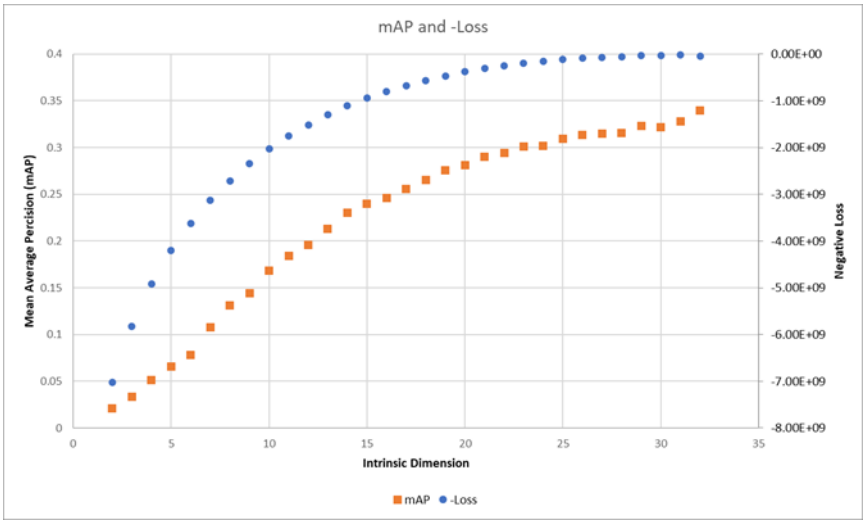
4.3 DeepMDS vs. BNN

The details below catalog an experiment in which we directly compare the Bayesian neural network to predecessor, the network outlined in [Gong, Boddeti, and Jain 2019]. We confirm the results of the prior study and also report on effectiveness of the Bayesian version of the same model.

4.3.1 DeepMDS

In Figure 6, we show the results of the DeepMDS process final negative loss and mAP values for reduction from 32 dimensions to lower dimensions between 2 and 31. Here we see a final mAP value for an intrinsic dimension of 19. [Gong, Boddeti, and Jain 2019] reported a 6% loss of mAP of 39% to 33%. Here we achieve the same reduction with an 8.8% loss of mean average precision (36.2% down to 27.4%).

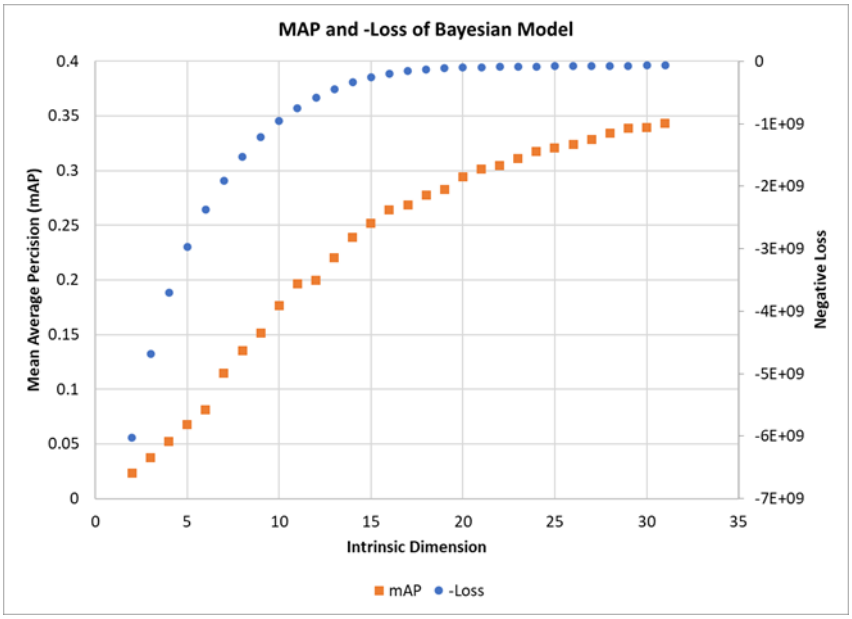
Figure 6: DeepMDS mAP and Loss



4.3.2 BNN

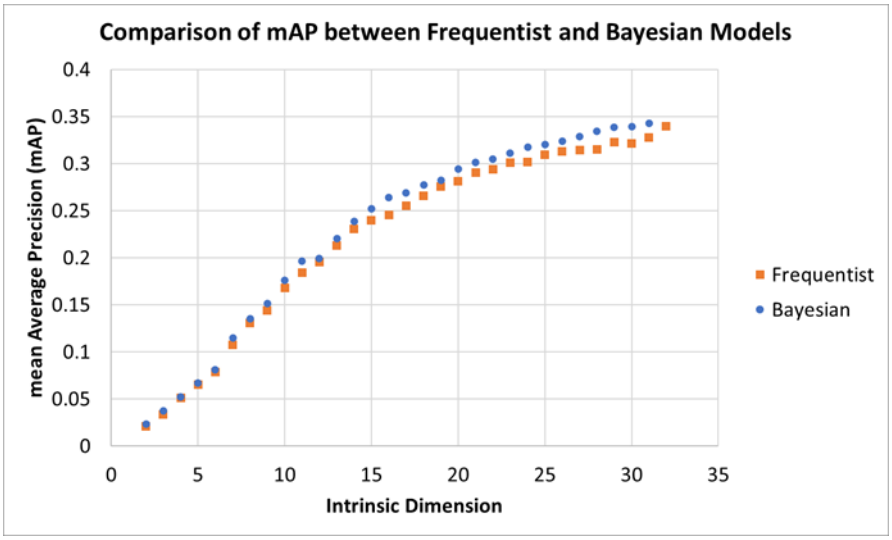
In Figure 7, we plot the map and negative loss for the Bayesian model seeing similar results. From the loss curve we can estimate the point in which we begin to see little change in loss for an increasing dimension suggesting an intrinsic dimension between 15 and 25.

Figure 7: Bayesian mAP and Loss



With both models run, we can directly compare the mean Average Precision of the two methods. In Figure 8 below, we see a small (about 1%) increase of accuracy for values around the intrinsic dimension for our Bayesian process.

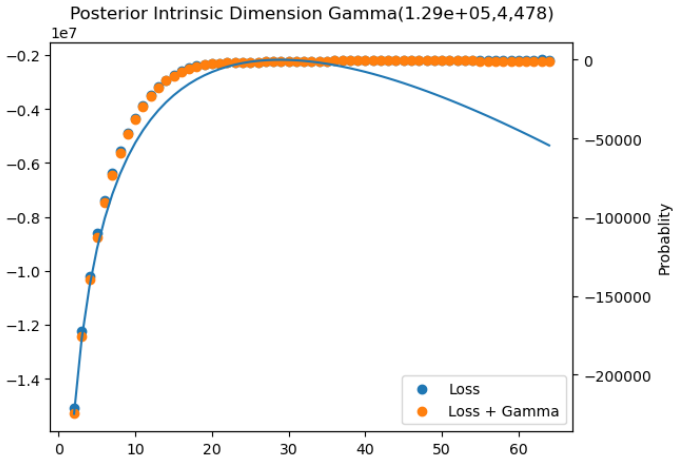
Figure 8: DeepMDS. vs. BNN Comparison of mAP



4.4 BNN with Prior ID Estimate

Figure 9 shows the transformation of the intrinsic dimension produced by the BNN when given the prior estimate from our Two NN ID Estimation. The original data in blue circles, and resting on top and slightly below is the transformed data in orange squares, suggesting nearly identical results.

Figure 9: Bayesian Neural Network with ID prior



5 Conclusions

5.1 Recreation of Baseline DeepMDS Version

Overall, the results by [Gong, Boddeti, and Jain 2019] were successfully reproduced. We achieved 512 to 19 dimensionality reduction with an 8.8% loss of mean average precision (36.2% down to 27.4%). This was done with minimal parameter tuning, which may explain the 2% difference in Gong's results.

5.2 Bayesian Version

In swapping out the loss function and neural network nodes for Bayesian equivalent versions we also achieved comparable results reducing dimensionality from 512 to 19 with 7.7% loss (36.2% down to 28.5%). This was done using identical hyperparameters to the baseline version. Thus, our Bayesian version attained similar if not a small improvement over the frequentest version. Moreover, the Bayesian version appears to have a small improvement at all dimensions from 15 and above. Our technique provides a range of intrinsic dimension accuracy suggesting an optimal intrinsic dimension of 15 to 24. Lastly, our process outputs a range of solutions which can quantify the uncertainty for a given intrinsic dimension, which is a notable improvement in itself. While our method showed modest gains in ImageNet, we suspect our model application would be more applicable in other settings.

5.3 BNN with Prior ID Estimate

Lastly, adding the TwoNN prior to our BNN estimation resulted in a very small shift in values, likely due to the overwhelming amount of data. This contained a slightly more pronounced maxima and a slightly narrower range of Intrinsic Dimensions.

6 Future Work

6.1 Transfer Learning

We would like to explore the concept of transfer learning as the training of the ImageNet data requires a large of amount of time. With transfer learning we expect that we can train a general form of dimensionality reduction such that given new categories we can include them into the lower dimensional manifold with minimal training.

6.2 ImageNet 1000

We seek to utilize the the full ImageNet library of 1000 categories, totaling over 1.3 million images to analysis how K categories impacts the ID.

6.3 Asymptotic Maximum Intrinsic Dimension

We wish to explore the question "As K categories go to infinity, is there an upper bound on the intrinsic dimension such that we can capture 99% of all imaged subjects."

REFERENCES

[Gong, Boddeti, and Jain 2019] Gong, s.,Boddeti, V. N., Jain, A. K. (2019). On the Intrinsic Dimensionality of Image Representation, IEEE Trans. Neural Networks Learn. Wiley, New York.

[Liu, Bhattacharya, and Maiti 2021] Liu, Z., Bhattacharya, S., Maiti, T. (2021). Variational Bayes Ensemble Learning Neural Networks With Compressed Feature Space, 2nd ed. Wiley, New York.

[Bennett, R. S. 1965] Bennett, R. S. (1965). Representation and Analysis of Signals Part XXI. The Intrinsic Dimensionality of Signal Collections, Johns Hopkins University, Baltimore.

[Rytgaard 1990] Rytgaard, R. (1990). Estimation in the Pareto Distribution, ASTIN Bulletin, Vol. 20, No. 2, Copenhagen

[Denti, et. al. 2022] Denti, F., Doimo, D., Laio, A., and Mira, A. (1990). The generalized ratios intrinsic dimension estimator, Nature Portfolio, Copenhagen.

[Bishop 2006] Bishop, C. M. (2006). Pattern Recognition and Machine Learning, Springer, New York.

[Loshchilov and Hutter 2017] Loshchilov, I. Hutter, F. (2017). Decoupled Weight Decay Regularization, Cornell University.

[Allegra, et. al. 2020] Allegra, M. Facco, E. Denti, F. Laio, A. Mira, A. (2017). Data segmentation based on the local intrinsic dimension, Nature.