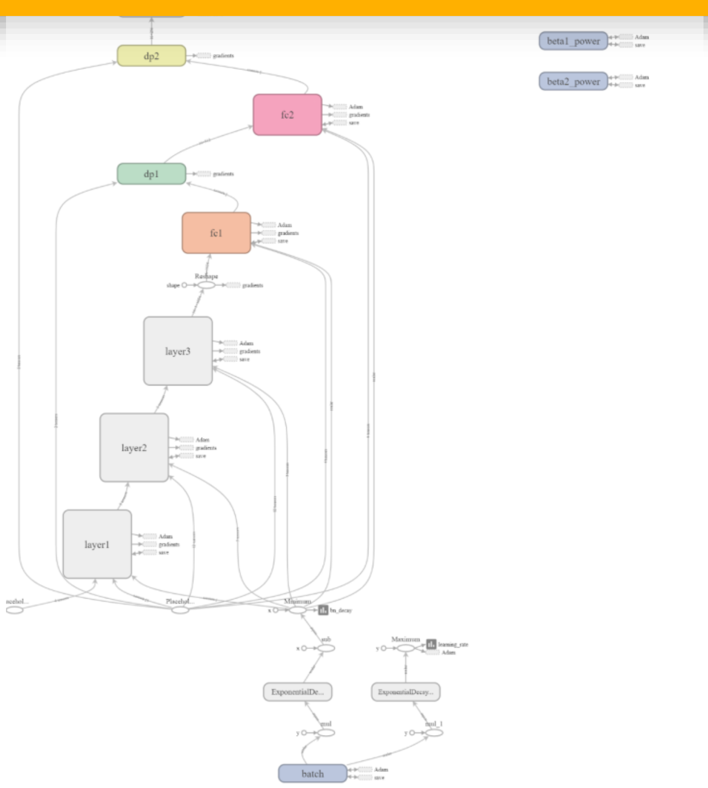
Bayesian Optimization of Pointnet

Laxman Singh (A0178223E)
Niranchana Periasamy (A0150379B)
Siew Yaw Hoong (A0178544U)
Tan Chee Wei (A0179723U)

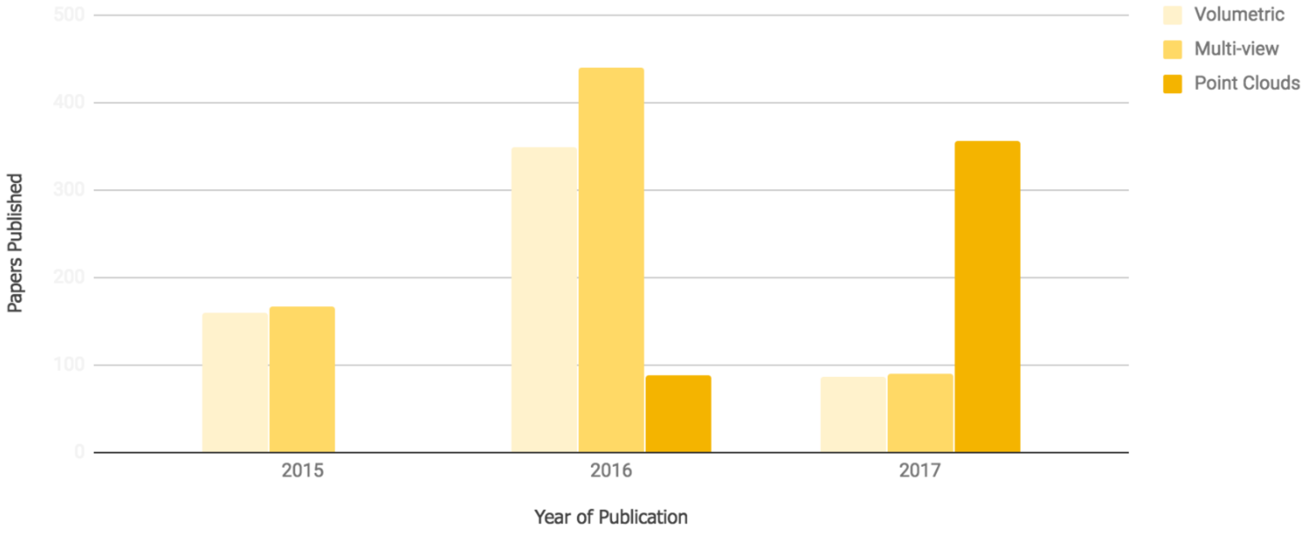
WHAT WE HAVE DONE

- Subject matter understanding
  - Point cloud
  - Pointnet
  - Tensorflow / Tensorboard
  - CNN
- Literature research
  - Pointnet
  - Pointnet++
  - 3DmFV
- Installation and code execution
  - Local CPU/GPU
  - Colab GPU/TPU
- Comparing results



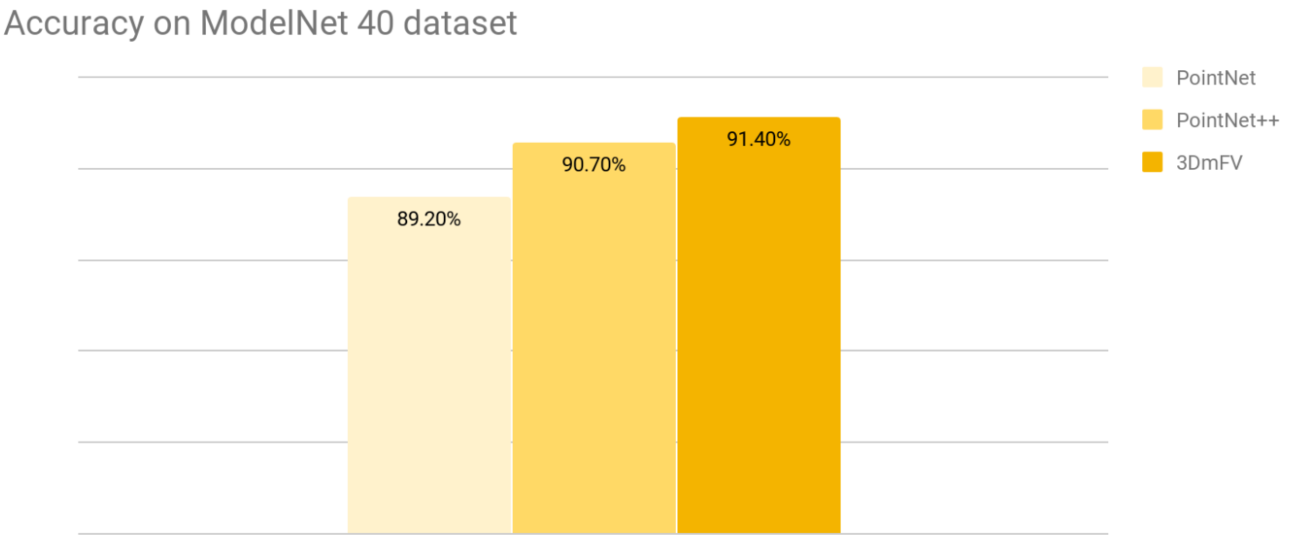
LITERATURE REVIEW

Point Cloud gained prominence since 2017 when point cloud scanning was widely used by scanners for converting multiple point cloud to a 3D model



Framework	Technique Used	Methodology
PointNet	• Point Classification • Point Segmentation • Semantic Segmentation	Due to point cloud's unstructured format & invariance to permutations, PointNet is proposed to apply deep learning on each point directly
PointNet++	• Point Classification • Point Segmentation	Essentially an hierarchical version of PointNet, introduced to capture fine-grained patterns and generalize complex scenes
3D Modified Fisher Vectors (3DmFV)	• Point Classification • Point Segmentation	To address severe accuracy vs memory size tradeoffs of PointNet, 3DmFV-Net architecture introduces hybrid method that combines discrete grid structure with continuous generalized Fisher vectors (feature aggregation)

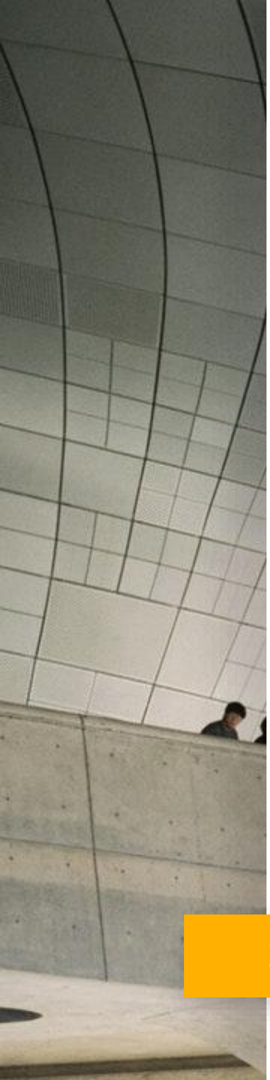
Model Results - Point Classification

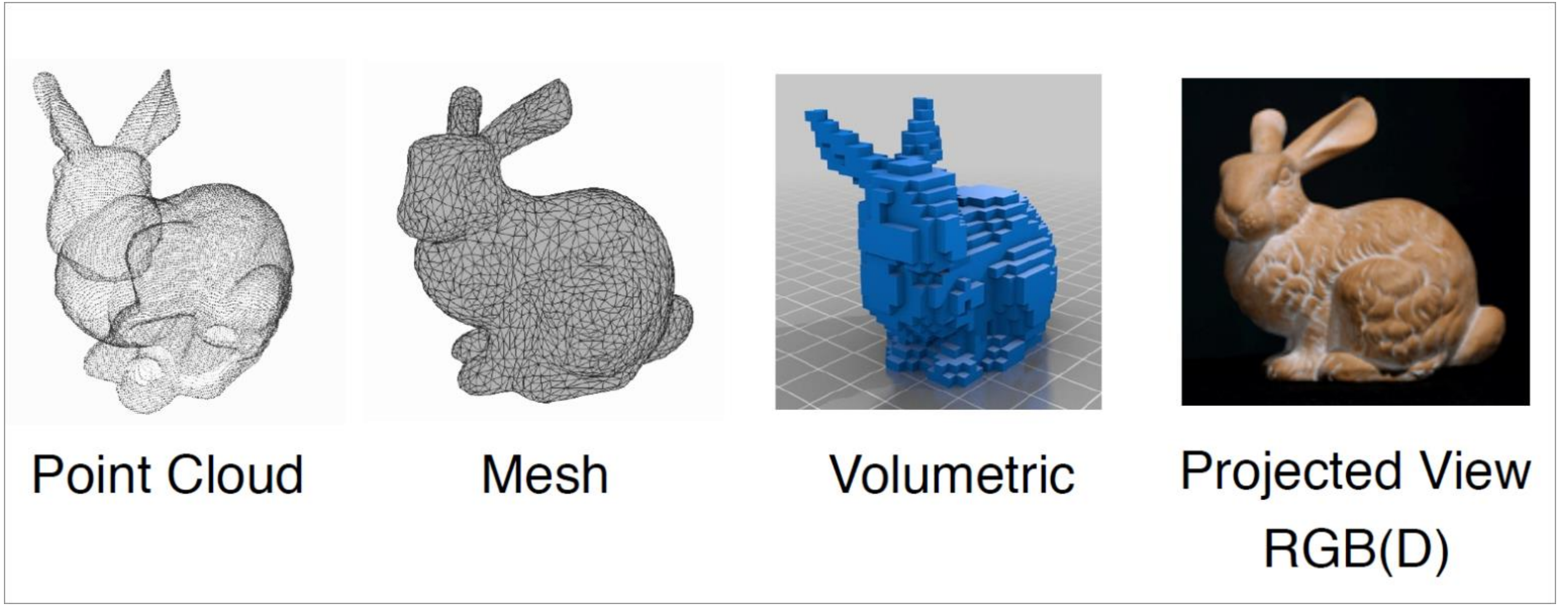


Research on optimizing model performance

Framework	Technique Used	Methodology
Revisiting small batch training for deep Neural Network	Mini-batch Stochastic Gradient Descent (SGD) optimization	• Propose small batch training as it is observed the best performance is obtained for mini-batch sizes between 2 and 32 which helps in improving accuracy • Although this increases the training time, we can Induce data parallelism to reduce the impact • Current training time - **12 hrs**
Algorithms for Hyper-Parameter Optimization	Sequential hyper-parameter optimisation algorithm	To overcome human intervention and drawbacks of random search algorithm in optimizing parameters of the model, TPE and sequential algorithms are proposed

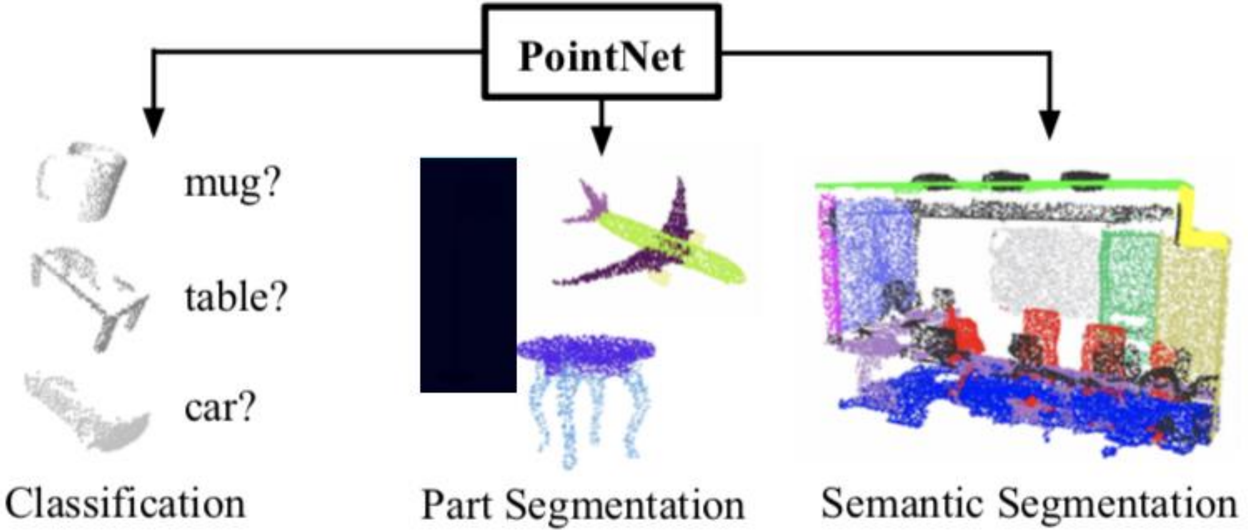
2. What is PointNet?





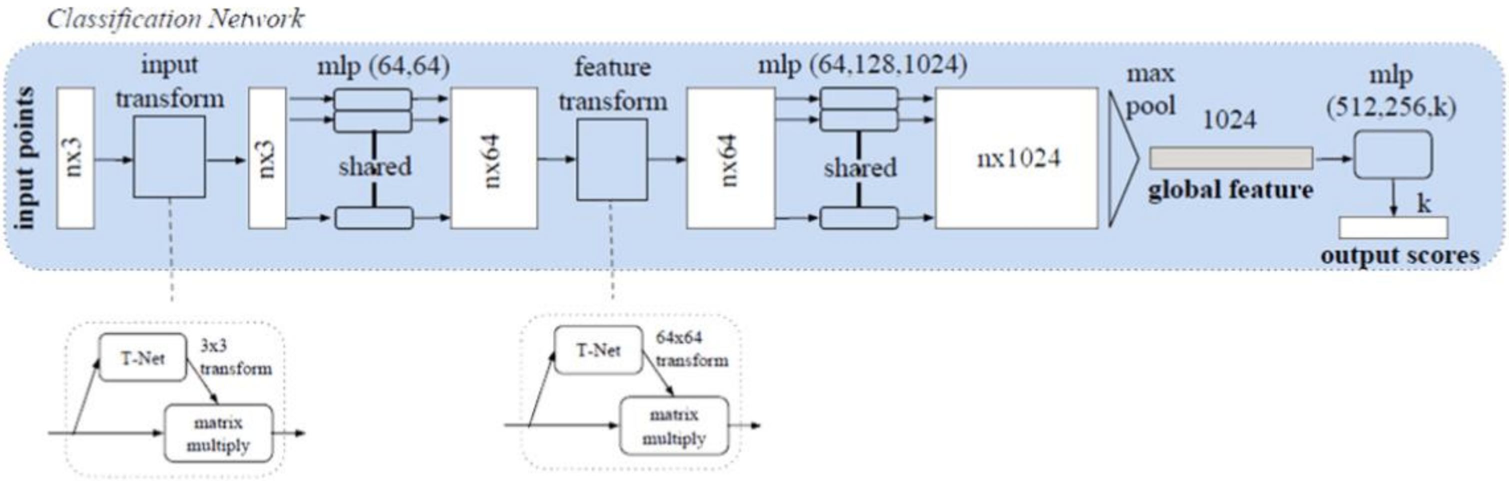
The conversion of a point cloud into a mesh, and then into a 3D model

Pointnet - Applications



- Classification: Classify an object
- Part Segmentation: Classifying part of an object
- Semantic segmentation: associating each pixel of an image with a class label

Pointnet - Classification Architecture

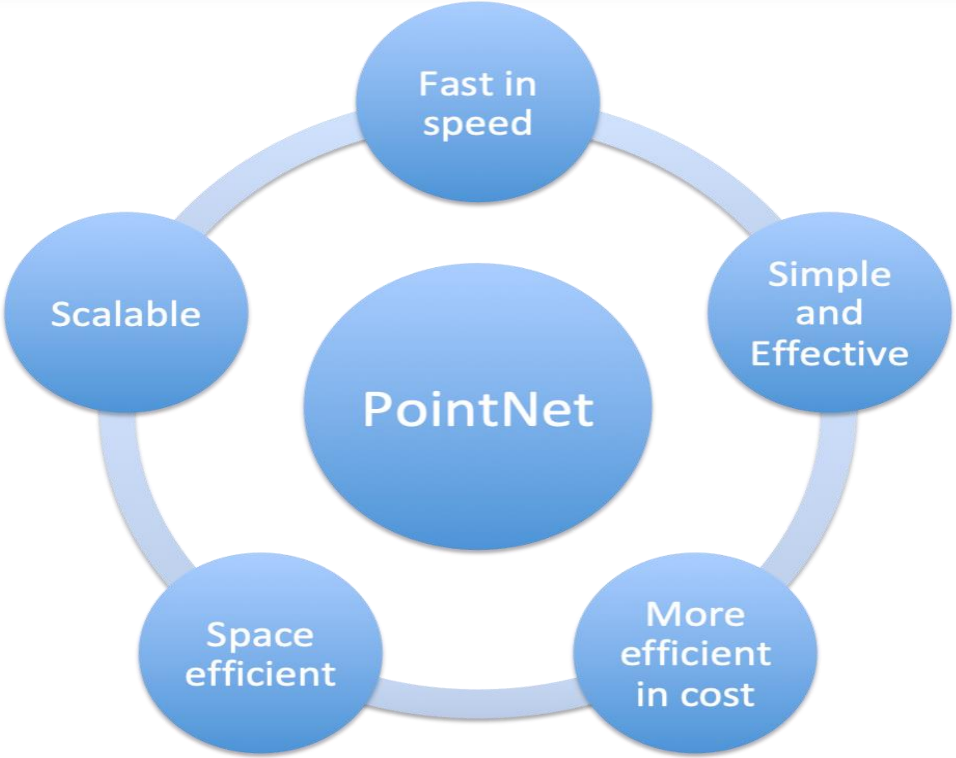


- Mini-PointNet - Takes raw input and regresses to 3*3 matrix.
    - MLP on each point (with shared weight)
    - Max pooling
    - Two fully connected layers with output sizes 512,256

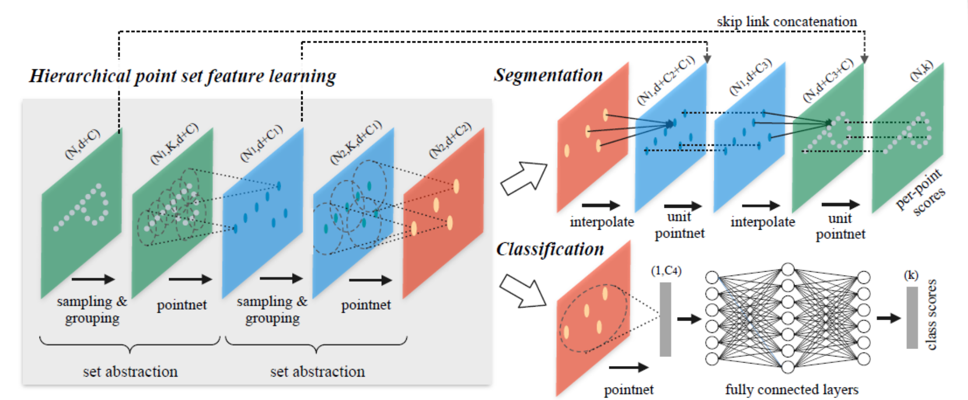
Pointnet - NN & Data Challenges

- **Neural Network**
  - Unstructured Data
  - Invariant to permutations
  - Number of point changes
- **Data Related**
  - Missing Data
  - Noise
  - Volume of Data
  - Rotation

Pointnet - Advantages

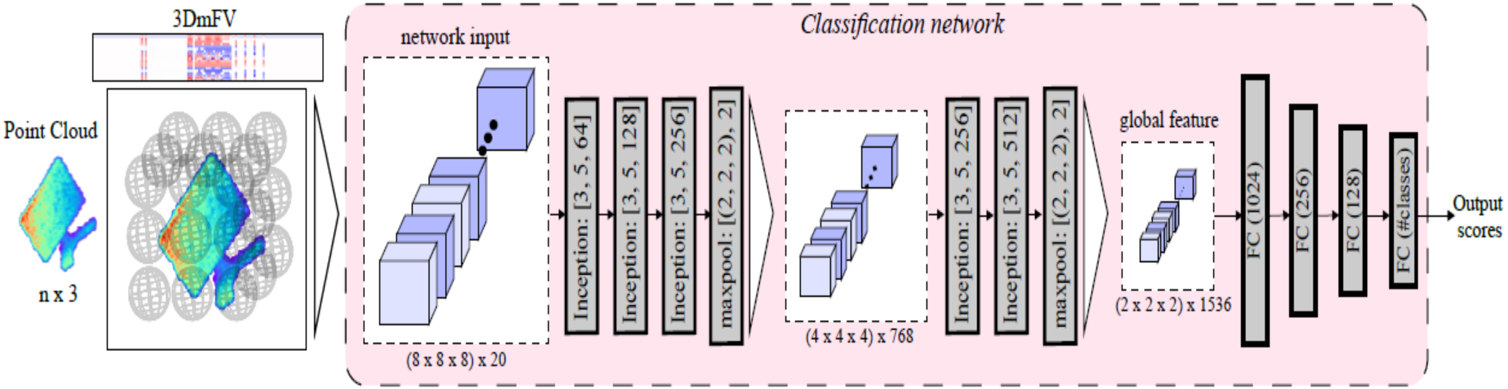


Pointnet++



- Pointnet++ is hierarchical version of PointNet
- Each layer has three sub stages: sampling, grouping, and PointNeting.
- First stage : Select centroids
- 2nd stage : Using surrounding neighbouring points, create multiple sub-point clouds.
- Pointnet is able to produce high dimension outputs using these multiple sub-point clouds.
- This process is repeated.

3DmFV

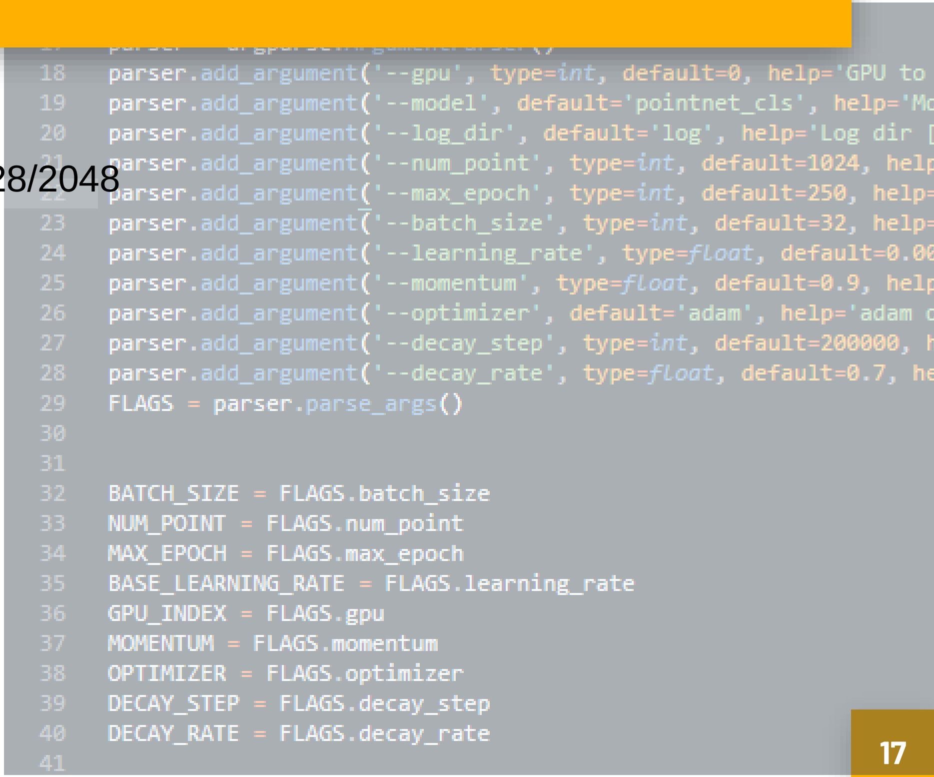


- New point cloud hybrid representation
- Representation describes points by their deviation from Gaussian Mixture Model(GMM)
- This keeps continuous properties of point cloud.
- It has better classification accuracy even with low resolution.

3. Framework Comparison

POINTNET MODEL PARAMETERS

- Batch size - 32
- Number of points - 256/512/1028/2048
- Max epoch - 250
- Learning rate - 0.001
- Momentum - 0.9
- Optimizer - Adam
- Decay step - 200,000
- Decay rate - 0.7



```
parser.add_argument('--gpu', type=int, default=0, help='GPU to
parser.add_argument('--model', default='pointnet_cls', help='Mo
parser.add_argument('--log_dir', default='log', help='Log dir [
parser.add_argument('--num_point', type=int, default=1024, help
parser.add_argument('--max_epoch', type=int, default=250, help=
parser.add_argument('--batch_size', type=int, default=32, help=
parser.add_argument('--learning_rate', type=float, default=0.00
parser.add_argument('--momentum', type=float, default=0.9, help
parser.add_argument('--optimizer', default='adam', help='adam o
parser.add_argument('--decay_step', type=int, default=200000, h
parser.add_argument('--decay_rate', type=float, default=0.7, he
FLAGS = parser.parse_args()


BATCH_SIZE = FLAGS.batch_size
NUM_POINT = FLAGS.num_point
MAX_EPOCH = FLAGS.max_epoch
BASE_LEARNING_RATE = FLAGS.learning_rate
GPU_INDEX = FLAGS.gpu
MOMENTUM = FLAGS.momentum
OPTIMIZER = FLAGS.optimizer
DECAY_STEP = FLAGS.decay_step
DECAY_RATE = FLAGS.decay_rate
```

MODEL COMPARISON

	Training			Evaluation			Published
	Mean Loss	Accuracy	Average Class Accuracy	Mean Loss	Accuracy	Average Class Accuracy	Accuracy
Pointnet	0.552	0.885	0.859	0.546	0.883	0.856	0.892
Pointnet++	0.451	0.895	0.872	0.453	0.895	0.873	0.907
3DmFV	0.497	0.896	0.868				0.914

4. Hyperparameter Optimisation

Yann LeCun
@ylecun

Following

Training with large minibatches is bad for your health.
More importantly, it's bad for your test error.
Friends dont let friends use minibatches larger than 32. arxiv.org/abs/1804.07612

5:00 AM - 27 Apr 2018

Ref: Revisiting small batch training for deep NN

- Small batch size improve performance and accuracy
- Best results with m=32, often as small as m=2 or m=4
- Cons: reduce computational parallelism available

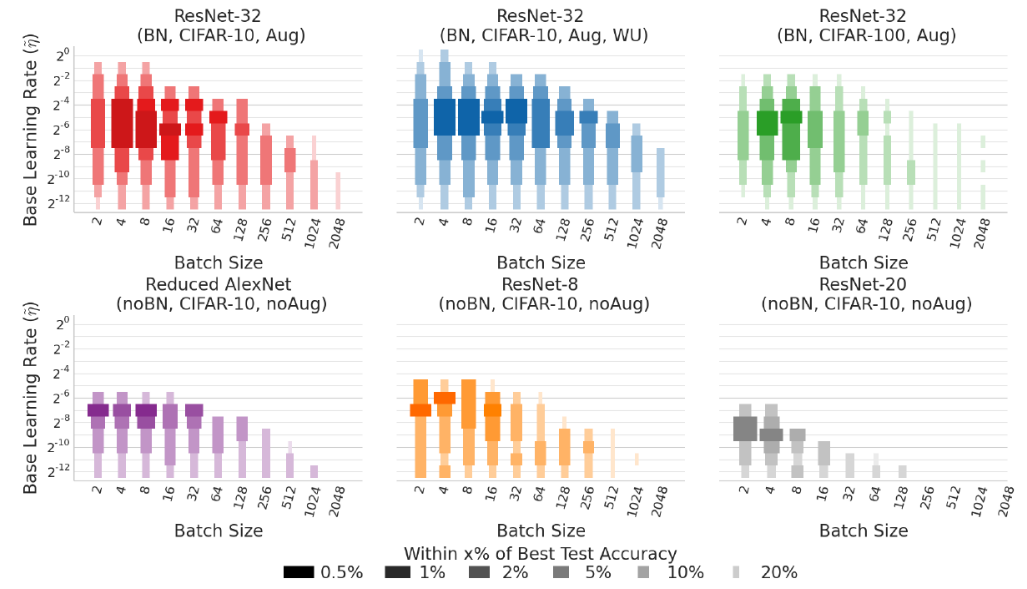
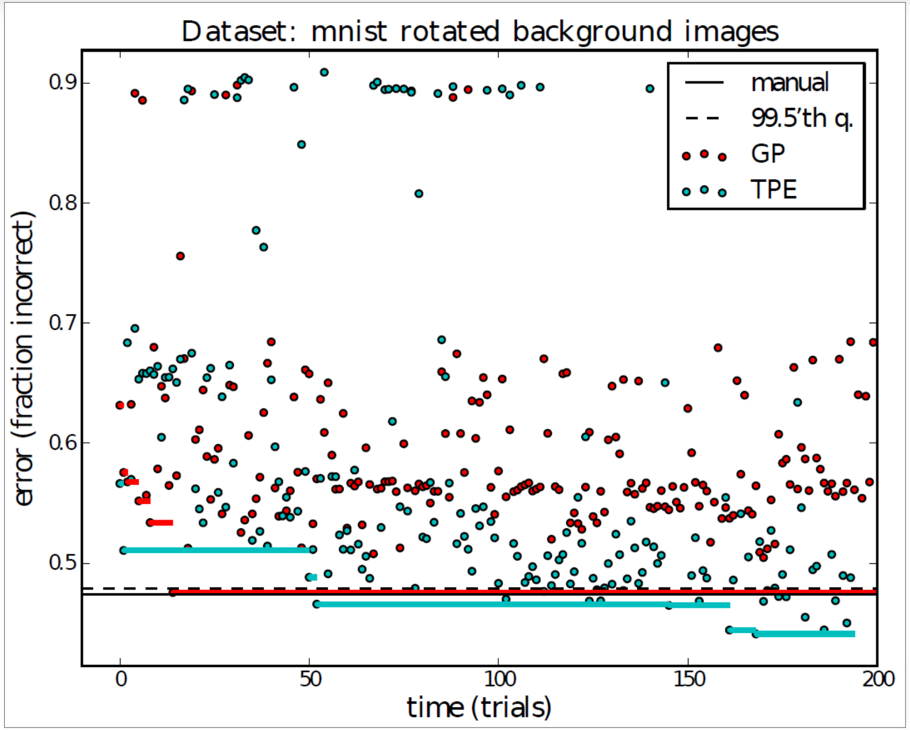


Figure 16: Summary of range of base learning rates $\tilde{\eta} = \eta/m$ that provide reliable convergence, for different network architectures. CIFAR-10 and CIFAR-100 datasets. (BN, noBN: with/without BN; Aug, noAug: with/without data augmentation; WU: with gradual warmup.)

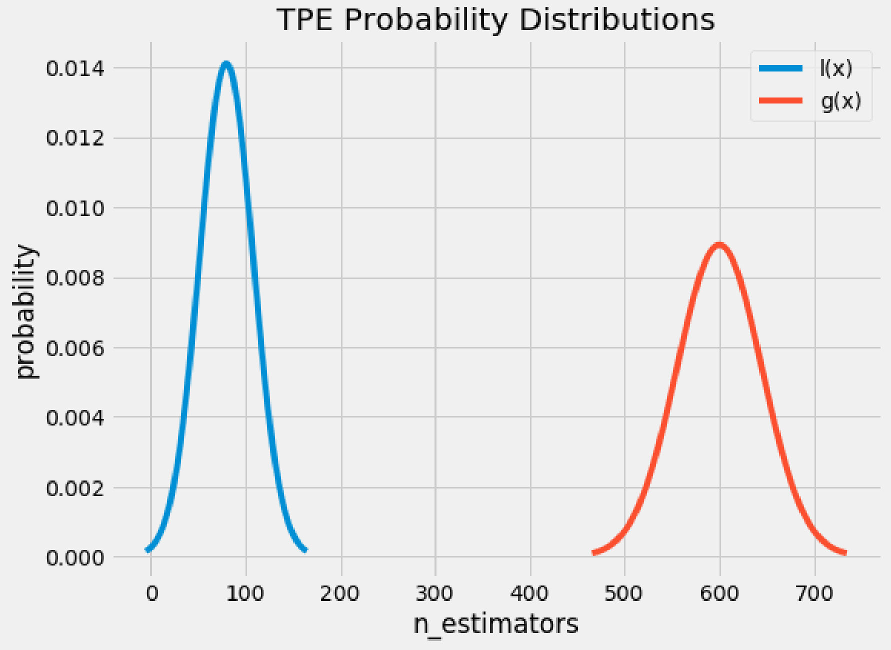
Ref:Algorithms for Hyper-Parameter Optimization

- Based on recent works, random search was shown to be more efficient than grid search.
- Tree-structured Parzen Estimator (TPE) outperforms manual and random search methods



Tree-structured Parzen Estimator (TPE)

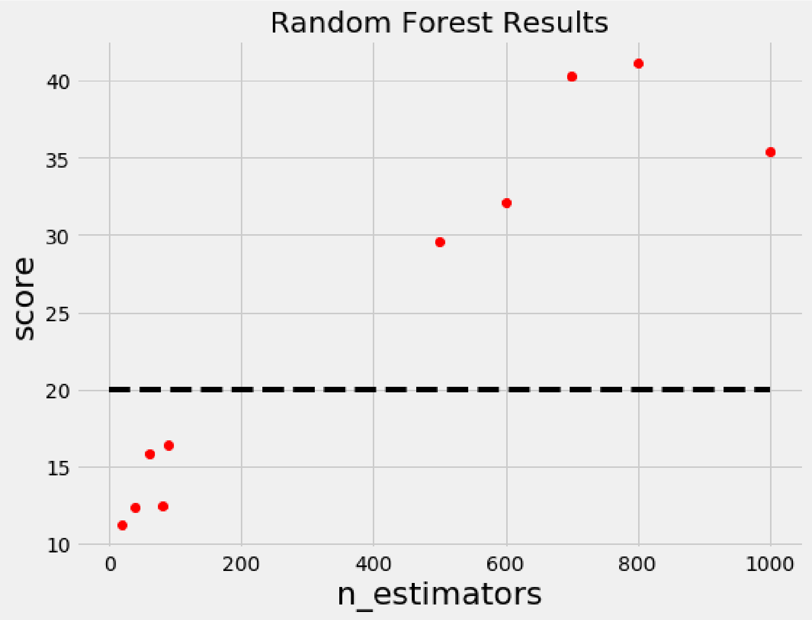
- Hyperparameters are split into two distributions based on the threshold, l(x) - less than threshold, g(x) - greater than threshold
- Draw sample hyperparameters from l(x) and evaluate them in terms of g(x)/l(x) and return the set that yields the highest value which corresponds to the greatest expected improvement (EI)



$$EI_{y^*}(x) = \frac{\gamma y^* \ell(x) - \ell(x)\int_{-\infty}^{y^*} p(y)dy}{\gamma \ell(x) + (1-\gamma) g(x)} \propto \left(\gamma + \frac{g(x)}{\ell(x)}(1-\gamma)\right)^{-1}$$

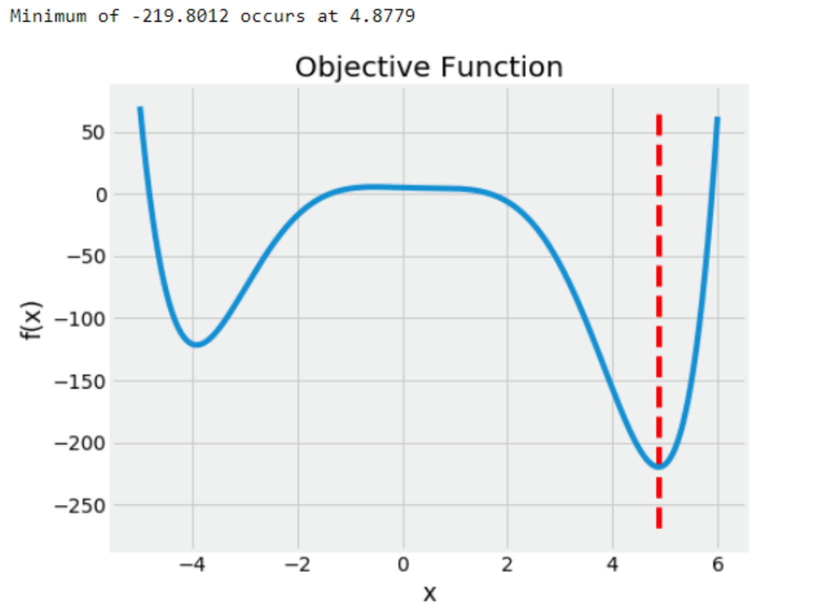
Bayesian Optimisation (BO)

- Optimise by building a probability model of the objective function that maps input values to a probability of a loss: p (loss | input values).
- The probability model, also called the surrogate or response surface, is easier to optimize than the actual objective function.
- Bayesian methods select the next values to evaluate by applying a criteria (usually Expected Improvement) to the surrogate



Bayesian Optimization (BO) - Cont

- Each time the algorithm proposes a new set of candidate hyperparameters, it evaluates them with the actual objective function and records the result in a pair (score, hyperparameters).
- These records form the history.
- The algorithm builds l(x) and g(x) using the history to come up with a probability model of the objective function that improves with each iteration.



Where’s the Result?

What’s Next?

Implement Hyperparameters Optimisation

Research into hyperopt framework and implement this on the Pointnet framework

Benchmark against default Model

Compare the result, analyze and document the changes.

Submit Final paper + Results

Fine tune the final code, and record the results and findings in a journal for submission.

THANKS!

Any questions?