MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Midterm Review
Project (P3/P4) help

3/11/2025

Session Agenda

- Reminder: P3 Due TODAY (March 11, 2025).
- P4 PoseCNN part available, start working on it NOW!!!

Today:

- Midterm review
- project help

Final Project Team SignUp Reminder

https://docs.google.com/spreadsheets/d/1FjWAjJ8p26xZmZaqsW4lew8H4iKe0FA78Q30eZ38g7A/edit?usp=sharing

Please sign up by Thursday March 13, 2025
After that, instructors will assign teams.

MidTerm

Tomorrow March 12, 2025 12PM - 1:30PM (in-class), 2246 CSRB

- Pen/Pencil and Paper – Please bring your own pen/pencil!
- 1 A4/Letter-size note sheet (front and back).
- No GenAI/phone/computer/internet

A mix of true/false, multiple choice/answer, and free response questions.
Total: 100 points (count as 10% of total grade - individual grade)

*Disclaimer

We put together this set of practice questions for your reference and practice benefits.

Not all questions covered today will appear on the exam - but still worth knowing. There could be also questions that are not covered today (but are covered in lectures/previous discussions/projects/quizzes etc.).

PyTorch

What are some of the key characteristics of PyTorch versus other machine learning frameworks?

PyTorch

What are some of the key characteristics of PyTorch versus other machine learning frameworks?

Ease of use and learning, dynamic computation graphs, high adaptation, geared towards research

Pytorch

What is the key data type (container) used in PyTorch?

Pytorch

What is the key data type (container) used in PyTorch?

Tensor

Pytorch

What are the two devices we use in class? What are their key characteristics?

Pytorch

What are the two devices we use in class? What are their key characteristics?

device = {'cpu','cuda'}

Core count: cuda~10000's, cpu~10's

Instruction set: cuda~simple, cpu~complex

Clockspeed: ~2-5GHz for both

Pytorch

What functions can be used to initialize tensors?

Pytorch

What functions can be used to initialize tensors?

torch.zeros( )
torch.ones( )
torch.arange( )
torch.rand( ) and torch.randn( )
torch.***_aslike( )

Pytorch

What are key parameters in initialization?

Pytorch

What are key parameters in initialization?

Size: torch.ones(x) vs. torch.ones(y)
Dimensionality: torch.ones(x) vs. torch.ones(x, y)
Datatype: torch.ones(..., dtype=float32)
Device: torch.ones(..., device='cuda')

Pytorch

How to initialize this tensor?

```
tensor([[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]], device='cuda:0')
```

Pytorch

How to initialize this tensor?

```
tensor([[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]], device='cuda:0')
```

```
A = torch.ones(3,5, device='cuda')
```

Pytorch

How to initialize this tensor?

```
tensor([[[0.5724, 0.8775, 0.7427, 0.8513, 0.6840, 0.0101],
         [0.5954, 0.8389, 0.5471, 0.9058, 0.4669, 0.4252],
         [0.5167, 0.0119, 0.7375, 0.4196, 0.0760, 0.7310],
         [0.7613, 0.7898, 0.6833, 0.5203, 0.6205, 0.3706]],

        [[0.4417, 0.7701, 0.3453, 0.3791, 0.4079, 0.0936],
         [0.0357, 0.8006, 0.0302, 0.0748, 0.3063, 0.2832],
         [0.0039, 0.6055, 0.5508, 0.4093, 0.7017, 0.4161],
         [0.2226, 0.7356, 0.4733, 0.5806, 0.2218, 0.1926]]])
```

Pytorch

How to initialize this tensor?

```
tensor([[[0.5724, 0.8775, 0.7427, 0.8513, 0.6840, 0.0101],
         [0.5954, 0.8389, 0.5471, 0.9058, 0.4669, 0.4252],
         [0.5167, 0.0119, 0.7375, 0.4196, 0.0760, 0.7310],
         [0.7613, 0.7898, 0.6833, 0.5203, 0.6205, 0.3706]],

        [[0.4417, 0.7701, 0.3453, 0.3791, 0.4079, 0.0936],
         [0.0357, 0.8006, 0.0302, 0.0748, 0.3063, 0.2832],
         [0.0039, 0.6055, 0.5508, 0.4093, 0.7017, 0.4161],
         [0.2226, 0.7356, 0.4733, 0.5806, 0.2218, 0.1926]]])
```

```
B = torch.rand(2,4,6)
```

Pytorch

Use `arange` to initialize:

```
C: tensor([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
D: tensor([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
E: tensor([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
F: tensor([100,  90,  80,  70,  60,  50,  40,  30,  20,  10])
```

Pytorch

Use `arange` to initialize:

```
C: tensor([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
D: tensor([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
E: tensor([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
F: tensor([100,  90,  80,  70,  60,  50,  40,  30,  20,  10])
```

```
C = torch.arange(start=0, end=10,step=1)
D = torch.arange(start=9, end=-1, step=-1)
E = torch.arange(0,20,2)
F = torch.arange(100, 0, -10)
```

Pytorch

How to select data from tensor?

Pytorch

How to select data from tensor?

Brackets and slicing. Axes are bypassed and/or implied via ':' operator.

Pytorch

Select as follows:

```
G: tensor([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
g: tensor(5)
```

Pytorch

Select as follows:

```
G: tensor([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
g: tensor(5)
```

```
g = G[5]
```

Pytorch

Select as follows:

```
H: tensor([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
        [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
        [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
        [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
        [40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
        [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
        [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
        [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
        [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
        [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
h: tensor(57)
```

Pytorch

Select as follows:

```
H: tensor([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
        [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
        [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
        [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
        [40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
        [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
        [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
        [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
        [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
        [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
h: tensor(57)
```

```
h = H[5,7]
```

Pytorch

Select as follows:

```
I: tensor([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
        18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
        36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
        54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71,
        72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
        90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
i: tensor([ 0,  3,  6,  9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51,
        54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99])
```

Pytorch

Select as follows:

```
I: tensor([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
        18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
        36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
        54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71,
        72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
        90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
i: tensor([ 0,  3,  6,  9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51,
        54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99])
```

```
index = t.arange(0,100,3)
i = I[index]
```

Pytorch

Select as follows:

```
J: tensor([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
        [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
        [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
        [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
        [40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
        [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
        [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
        [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
        [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
        [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
j: tensor([ 0, 11, 22, 33, 44, 55, 66, 77, 88, 99])
```

Pytorch

Select as follows:

```
J: tensor([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
        [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
        [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
        [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
        [40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
        [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
        [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
        [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
        [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
        [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
j: tensor([ 0, 11, 22, 33, 44, 55, 66, 77, 88, 99])
```

```
index = t.arange(10)
j = J[index,index]
```

Pytorch

What function to find the max/min? What about the location of the max/min?

Pytorch

What function to find the max/min? What about the location of the max/min?

We use torch.min()/max()/argmin()/argmax().

Pytorch

What are one-hot tensors? How to create them?

```
L: tensor([0.1249, 0.9227, 0.3419, 0.5514, 0.1995])
l: tensor([0., 1., 0., 0., 0.])
```

Pytorch

What are one-hot tensors? How to create them?

```
L: tensor([0.1249, 0.9227, 0.3419, 0.5514, 0.1995])
l: tensor([0., 1., 0., 0., 0.])
```

```
l = t.zeros_like(L)
l[L.argmax()] = 1
```

Pytorch

Practice min/max:

```
K: tensor([0.8180, 0.8177, 0.4460, 0.3086, 0.8958])
k: tensor(0.8958)
k_arg: tensor(4)
```

Pytorch

Practice min/max:

```
K: tensor([0.8180, 0.8177, 0.4460, 0.3086, 0.8958])
k: tensor(0.8958)
k_arg: tensor(4)
```

```
k = K.max()
k_arg = K.argmax()
```

Pytorch

How to sum tensors in different dimensions?

```
M: tensor([[ 0,  1,  2,  3,  4],
        [ 5,  6,  7,  8,  9],
        [10, 11, 12, 13, 14]])
tensor(105)
tensor([15, 18, 21, 24, 27])
tensor([10, 35, 60])
```

Pytorch

How to sum tensors in different dimensions?

```
M: tensor([[ 0,  1,  2,  3,  4],
        [ 5,  6,  7,  8,  9],
        [10, 11, 12, 13, 14]])
tensor(105)
tensor([15, 18, 21, 24, 27])
tensor([10, 35, 60])
```

```
print(f'M: ' + str(m))
print(M.sum())
print(M.sum(dim=0))
print(M.sum(dim=1))
```

Pytorch

What are some of the fundamental functions used?

Pytorch

What are some of the fundamental functions used?

torch.cos(), torch.sin(), torch.abs(), torch.exp(), torch.log(), torch.floor(), torch.pow()

Pytorch

This snippet from Sydney's code is too slow. What are two things she can do?

```
print(
    torch.randn(10000, 10000, device="cpu", dtype=torch.float32).mm(
        torch.randn(10000, 10000, device="cpu", dtype=torch.float32)
        ).mean())
```

Pytorch

This snippet from Sydney's code is too slow. What are two things she can do?

Change the device to 'cuda'

```
print(
    torch.randn(10000, 10000, device="cpu", dtype=torch.float32).mm(
        torch.randn(10000, 10000, device="cpu", dtype=torch.float32)
    ).mean())
```

Optimization

Which of the following is a condition for two datasets in R^d to be linearly separable? Select all that apply.

a. The datasets contain more than *d* samples
b. The convex hulls of the datasets do not intersect
c. The datasets are normally distributed
d. There exists a hyperplane in R^d that perfectly separates the data
e. The datasets can be separated by an n^{th} order polynomial for $n > 2$

K-Nearest Neighbors

Suppose for a balanced dataset for binary classification (e.g., positive and negative examples are approximately 50% each for both train and test set) with n examples we run classification using K-nearest neighbors, with K=n. For simplicity, we assume that K is an odd number to make tie-breaking unambiguous. In this case, K-nearest neighbors is: (Select all correct options.)

a. n is a good choice for K.
b. The accuracy of this model would be higher than 90% over the test set.
c. K=n leads to a high bias.
d. Leads to a classifier that ignores the input.

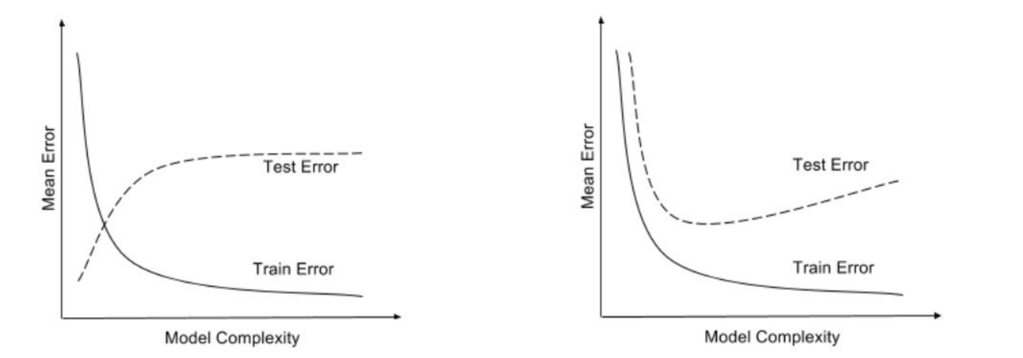
SVM

The farthest examples from the decision boundary of a dataset are called 'support vectors'. (Assume the dataset has extremely large number of examples.)

- True
- False

Training Neural Networks

Select one: Say you plot the train and test errors as a function of the model complexity. Which of the following two plots is your plot expected to look like?



(a) (b)

Assume we have a fully-connected neural network with 1 hidden layer with ReLU activations for binary classification. Which of the following statements are true about the behavior of the network? (*Select* <u>*all*</u> *correct options.*)

- Adding more layer sometimes perform worse than shallow networks.
- This model will have a non-linear decision boundary.
- The total training time is always the fastest for the smallest possible batch size since each gradient step takes less time.
- Multiplying all the weights and biases in the network by a factor of 10 after training the network will not change its classification accuracy.

Select all that apply: Which of the following statements about k-NN models is/are always true?

- Decreasing k makes a k-NN model have simpler or less complex decision boundaries.
- Increasing k makes a k-NN model less sensitive to outliers.
- k-NNs can be applied to classification problems but not regression problems.
- k-NNs can be applied to datasets with real-valued features but not categorical features.
- None of the above

Overfitting is characterized by:

- High variance and low bias
- Low variance and low bias
- Low variance and high bias
- High variance and high bias

Select all that are correct

- The gradient of the loss will always be 0 or close to 0 at a minimum
- The gradient of the loss may be 0 or close to 0 at a minimum
- The gradient of the loss may have large magnitude at a minimum
- If the gradient is not 0 at a minimum, it must be a local minimum

Logistic regression learns a non-linear decision boundary because the logistic function is non-linear.

- True
- False

We are given the function Y = F(G(H(X))), where Y and X are vectors, and G and H also compute vector outputs.

Select the correct formula for the derivative of Y w.r.t. X. We use the notation ∇_X(Y) to represent the derivative of Y w.r.t X.

- ∇_X(H) ∇_H(G) ∇_G(F)
- ∇_G(F) ∇_H(G) ∇_X(H)
- Both are correct

When initializing weights in a fully connected Neural Network, we should set the weight to 0 in order to preserve symmetry across all neurons.

- True
- False

Any multi-layer neural network with linear activation functions for all hidden layers can be represented as a neural network without any hidden layer.

- True
- False

Batch norm at any neuron is a vector operation over all the inputs in a minibatch

- True
- False

Two bounding boxes have areas of 100 pixels and 150 pixels, with an overlapping region of 50 pixels. What is the IoU between them?

A. 0.25
B. 0.33
C. 0.4
D. 0.5

Worth knowing how to compute IoU!!!

A model detects 15 objects, out of which 12 are correct, and 3 are incorrect. Additionally, the model fails to detect 6 objects. Calculate the precision and recall values.

A. Precision = 0.66, Recall = 0.8
B. Precision = 0.75, Recall = 0.5
C. Precision = 0.8, Recall = 0.66
D. Precision = 0.5, Precision = 0.75

Worth knowing how to compute Precision and Recall!!!

Note on Precision and Recall

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

In Object Detection:

- What is True Positive?
- What is False Positive?
- What is False Negative?

Q: Will changing IoU threshold change precision and recall value?

Gradients: Neural Network

Neural network layers:

- **Input layer:** 1 feature (x)
- **Hidden layer:** 1 neuron, sigmoid activation
- **Output layer:** 1 neuron

Gradients: Forward Pass

Forward pass calculations:

- **Hidden layer** $h = \sigma(w_h \bullet x)$
- **Sigmoid activation** $\sigma(z) = \frac{1}{1+e^{-z}}$
- **Output** $y = w_y \bullet h$
- **Loss** $L = \frac{1}{2}(y - \hat{y})^2$

Gradients: Question 1

Compute gradient of loss with respect to weights:

a) $\partial L / \partial w_y =$

b) $\partial h / \partial w_h =$

Gradients: Question 1

Compute gradient of loss with respect to weights:

a) $\partial L / \partial w_y = (y-\hat{y})(h)$

b) $\partial h / \partial w_h = (\sigma)(w_h x)(1-\sigma(w_h x))(x)$

Gradients: Question 2

Based on these gradients, will this produce a vanishing or exploding gradient problem? Why?

Try calculating the gradient with these parameters:
$x=0.5$, $w_h=10$, $y=0.8$, $\hat{y}=1.0$

Gradients: Question 2

Based on these gradients, will this produce a vanishing or exploding gradient problem? Why?

Vanishing (sigmoid leads to small derivatives for large or small weights)

Try calculating the gradient with these parameters:

$x=0.5$, $w_h=10$, $y=0.8$, $\hat{y}=1.0$

$h=\sigma(10*0.5)=\sigma(5)\approx 0.9933$

$\sigma'(5)=0.9933*(1-0.9933)\approx 0.0066$

Gradients: Leaky ReLU

Leaky ReLU is an activation function defined as:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

Gradients: Question 3

Complete the following function definition

```
def leaky_relu(x, alpha=0.01):
```

Gradients: Question 3

Complete the following function definition

```
def leaky_relu(x, alpha=0.01):
    return torch.where(x>0, x, alpha*x)
```

Gradients: Question 4

Why might Leaky ReLU address the vanishing gradient problem?

Gradients: Question 4

Why might Leaky ReLU address the vanishing gradient problem?

Avoids saturation at extreme values

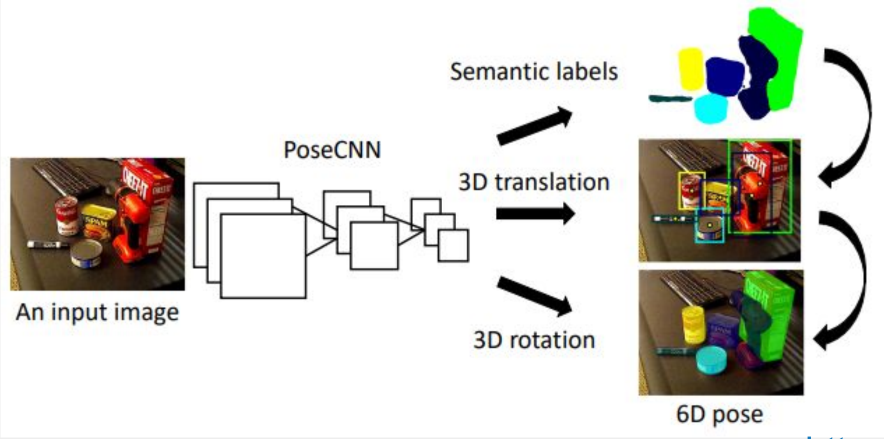
PoseCNN: Question 1

PoseCNN splits the task of acquiring a 6DOF pose of an object into which 3 subtasks?

https://arxiv.org/pdf/1711.00199

PoseCNN: Question 1

PoseCNN splits the task of acquiring a 6DOF pose of an object into which 3 subtasks?



https://arxiv.org/pdf/1711.00199

PoseCNN: Question 2

Before splitting into the 3 tasks mentioned above, what task can networks like PoseCNN perform that are shared among the subtasks?

https://arxiv.org/pdf/1711.00199

PoseCNN: Question 2

Before splitting into the 3 tasks mentioned above, what task can networks like PoseCNN perform that are shared among the subtasks?

Feature Extraction - 13 Convolution + ReLu layers, 4 Max Pooling layers

https://arxiv.org/pdf/1711.00199

PoseCNN: Question 3

The PoseCNN network's feature extraction layer begins pretrained on the VGG16 network using the ImageNet dataset. What are the benefits of doing so?

https://arxiv.org/pdf/1711.00199

PoseCNN: Question 3

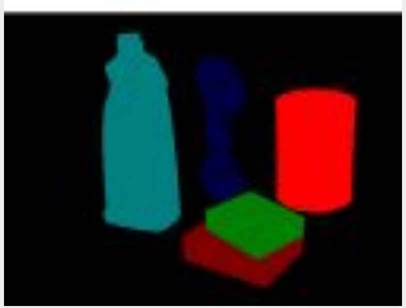
The PoseCNN network's feature extraction layer begins pretrained on the VGG16 network using the ImageNet dataset. What are the benefits of doing so?

Starting with a pretrained model, even if it's of a different model or dataset, can improve performance with a drastically reduce training time.

https://arxiv.org/pdf/1711.00199

PoseCNN: Question 4

Without knowing the orientation of an object, how might a network like PoseCNN calculate the center of the object?

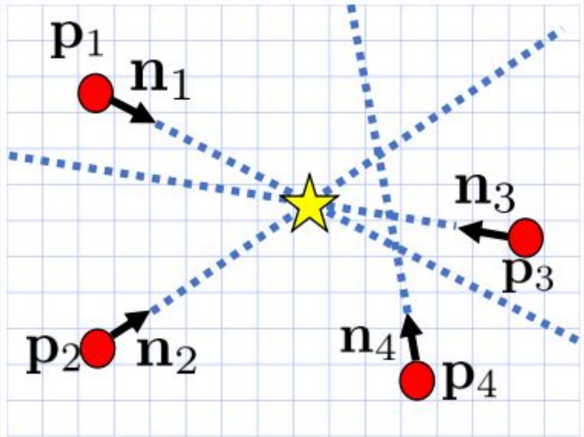


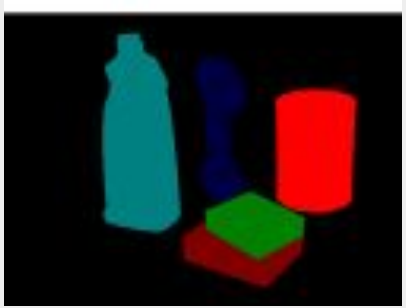
https://arxiv.org/pdf/1711.00199

PoseCNN: Question 4

Without knowing the orientation of an object, how might a network like PoseCNN calculate the center of the object?

A Hough Voting Layer





https://arxiv.org/pdf/1711.00199