How to apply Deep Learning to 3D objects

The Goal

- **First Part(General Strategy):** Get knowledge about how to approach making a deep learning product
- **Second Part(Specific case):** Get knowledge about deep learning applied to 3D objects

Table of Contents

Strategy

Reference: https://www.iconfinder.com/

Table of Contents

Specific Case

HELLO!

I am Masaya Ohgushi

I work at Kabuku Inc.

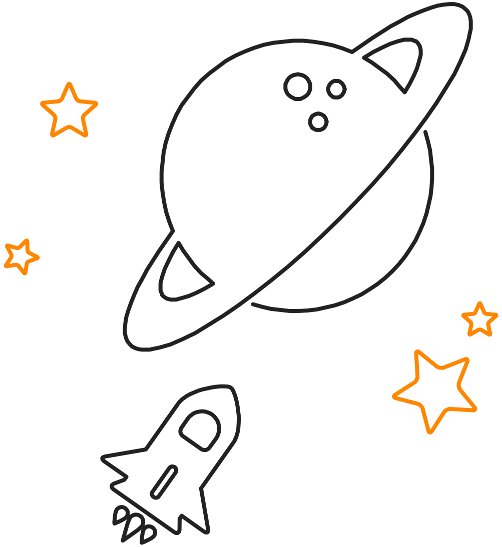
I am an image processing developer.

Twitter: @SnowGushiGit

Kabuku Inc.

- On-demand manufacturing service
  - Receive 3D data to manufacture by using 3D printers and others

Strategy



Strategy

Find the Right problem

Would you like to use deep learning to solve your problem?

Find the right problem

- What are the best cases to use Deep learning?
  - Most cases
    - Image processing
    - Speech recognition
  - Some cases
    - Natural language processing
    - Time series analysis

Right problem

- What are the worst cases to use Deep learning?
  - Not enough data
  - Can’t prepare a pre-train model
  - Need 100% accuracy

Strategy

Find the right method

How can you find the best way to solve your problem using deep learning?

Let’s search Google !!

Not possible!!

Find the Right method

- How to research the best deep learning solution? (In my case)
  - Google scholar
    - It is a paper search engine
      - You can find the following
        - Good methods
        - Good keywords
        - Which university laboratories know about this problem

Find the Right method

- How to research the best deep learning solution? (In my case)
  - University laboratory sites
    - It is possible to get data
    - It is possible to get code
  - GitXiv
    - You can find papers and code
  - Follow twitter users
    - It is possible to get the latest information

Find the Right method

- How to research the best deep learning solution? (In my case)
  - Book
    - You can get well structured knowledge
  - ArXiv
    - You can find the latest methods
  - Github
    - You can find code
  - Google
    - If you already know a good keyword !!

Strategy

Keep rechallenging

You gathered a lot of training data !!

Now let’s train using the full data set.

Not possible!!

Keep rechallenging

- Keep tries small
  - If you get a lot of data, first do the following things.
    - Prepare a small data set
      - Check module works correctly
    - Prepare an easy to verify training data set
      - Most models can be trained with data such as “mnist” , you have to check it works

Reference: https://www.tensorflow.org/get_started/mnist/beginners

Keep rechallenging

- A lot of challenges
  - There are no obvious methods to improve accuracy
    - You have to check the results
      - If training and validation accuracies don’t improve, you have to stop it
      - Check the results by visual boards such as TensorBoard
    - You have to increasing challenge times by improving the calculation speed
      - Using GPU
      - Optimize CPU

Strategy

Focus

Deep learning has a lot of methods to improve accuracy

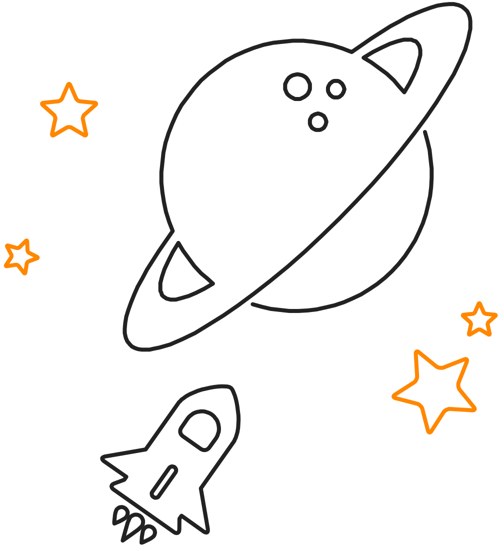
- *Model*
  - *How deep*
  - *How it is structured*
  - *Adjusting the hyper parameters*
- *Preprocess data*
  - *Data augmentation (If using Graphical data)*
- *Optimizer*
  - *SGD, Adam, etc*

Focus

- Focus on the right problem
  - Depends on your situation
    - Enough computation resource and enough data
      - Try a deep and complex model
    - Enough computation resource, but not enough data
      - Find a good pre-train model
      - Focus on the preprocess such as data augmentation

Focus

- Focus on the right problem
  - Depending on your situation
    - Not enough computation resource or data
      - Consider other ways to solve your problem
        - Logistic Regression, SVM, Random Forest
      - Deep learning probably isn't the best choice



Specific Case

Deep Learning applied to 3D objects

Specific Case

Deep Learning Model: VoxNet

There are a lot of deep learning models…

How to choose one

Deep Learning Model: VoxNet

- In my case, I considered 3 things
  - Resource
    - Computation resources
    - Human resources
  - Performance
    - accuracy
  - Speed
    - Speed of development

Deep Learning Model: VoxNet

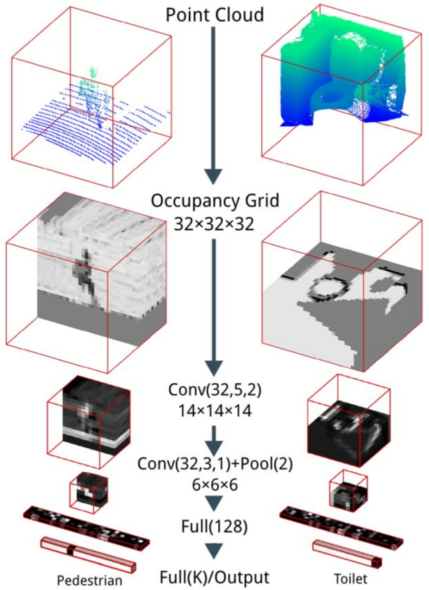


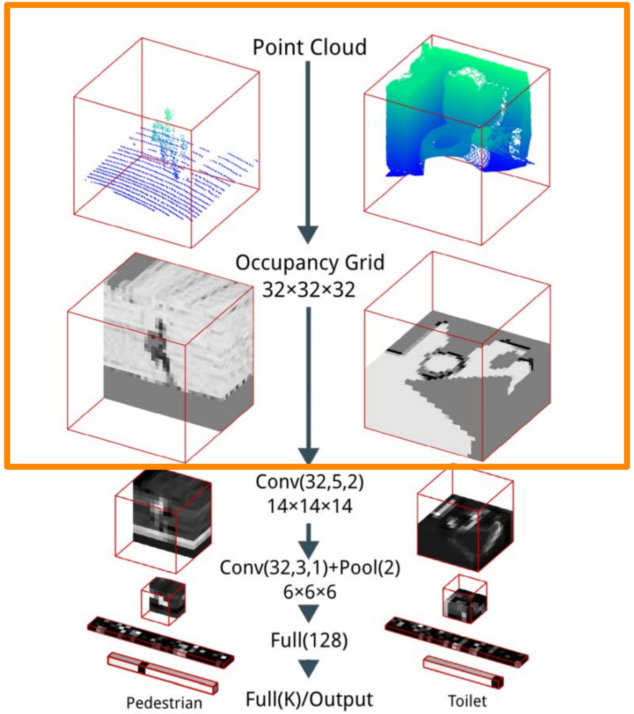
Fig. 1. The VoxNet Architecture. $Conv(f, d, s)$ indicates f filters of size d and at stride s, $Pool(m)$ indicates pooling with area m, and $Full(n)$ indicates fully connected layer with n outputs. We show inputs, example feature maps, and predicted outputs for two instances from our experiments. The point cloud on the left is from LiDAR and is part of the Sydney Urban Objects dataset [4]. The point cloud on the right is from RGBD and is part of NYUv2 [5]. We use cross sections for visualization purposes.

Reference: http://ri.cmu.edu/pub_files/2015/9/voxnet_maturana_scherer_iros15.pdf

Deep Learning Model: VoxNet

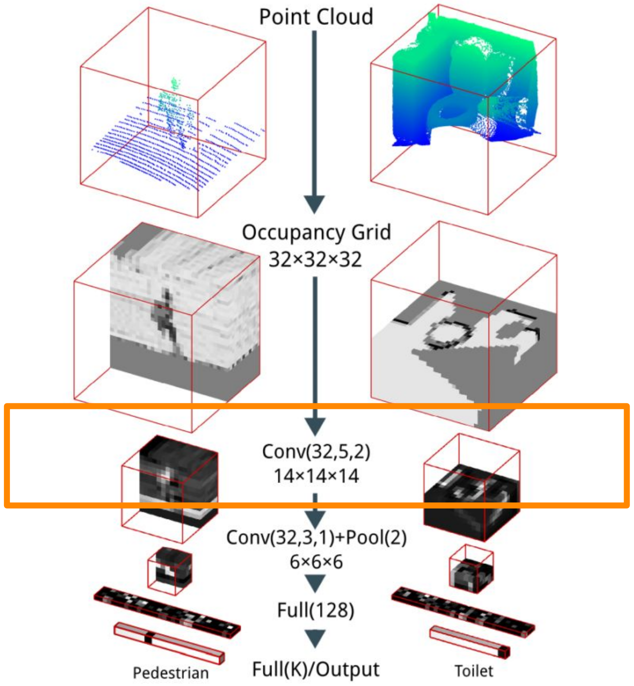
- VoxNet Advantage
  - Resource
    - Computation resources
      - Good
        - Memory 32GB (In my environment)
        - GPU GeForce GTX 1080 (In my environment)
  - Performance
    - Accuracy
      - 83 % accuracy (Top model 95 %)
  - Speed
    - Open source, simple code

Deep Learning Model: VoxNet

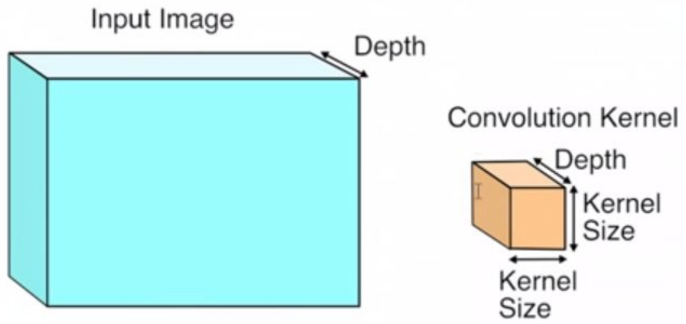


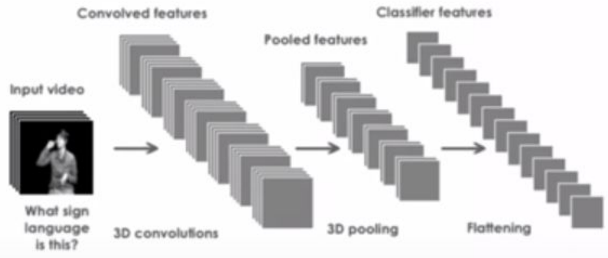
- Voxelize
  - Maps 3D data to a 32 * 32 * 32 voxel
  - Reduce data size

Deep Learning Model: VoxNet (3D CNN 3D objects)



- Convolution 3D

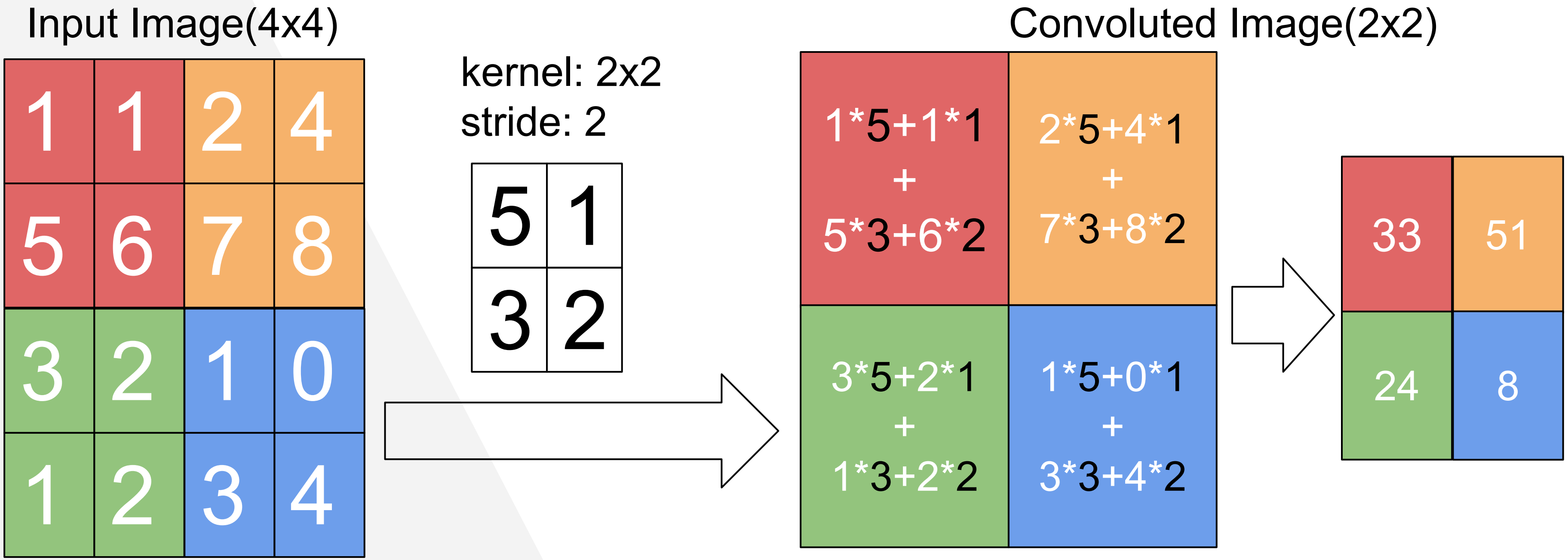




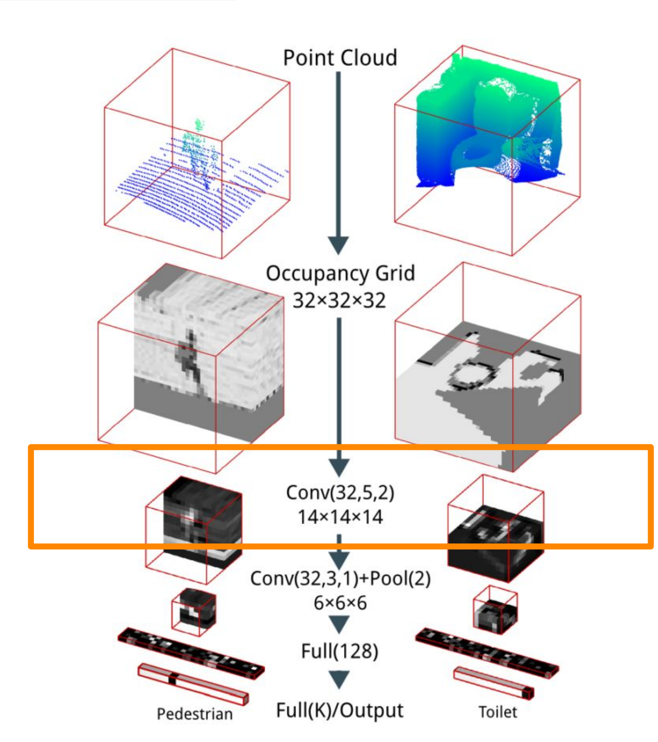
Reference: https://www.youtube.com/watch?v=ecbeIRVqD7g

Deep Learning Model: VoxNet (3D CNN 3D objects)

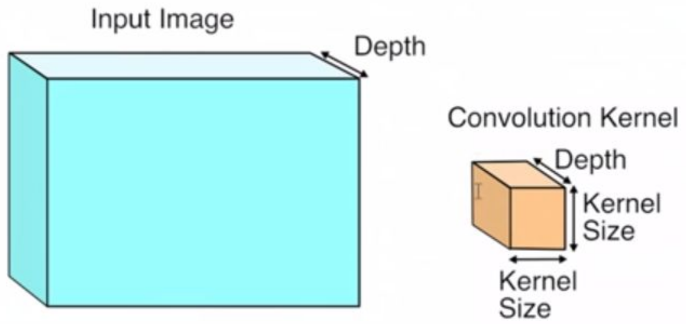
Convolution 2D

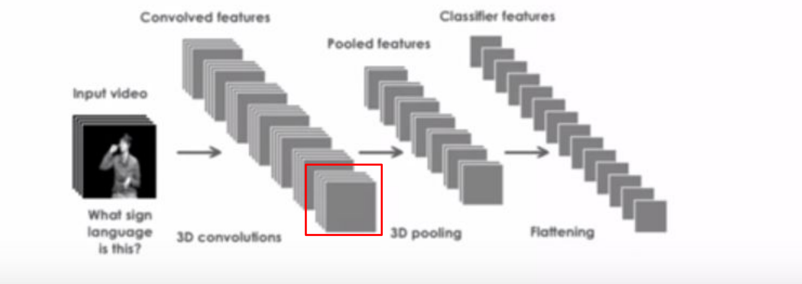


Deep Learning Model: VoxNet (3D CNN 3D objects)

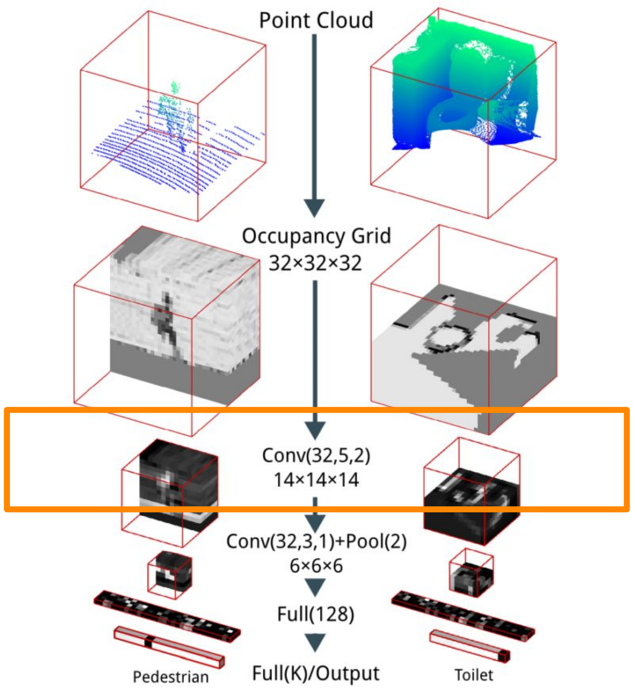


- Convolution 3D

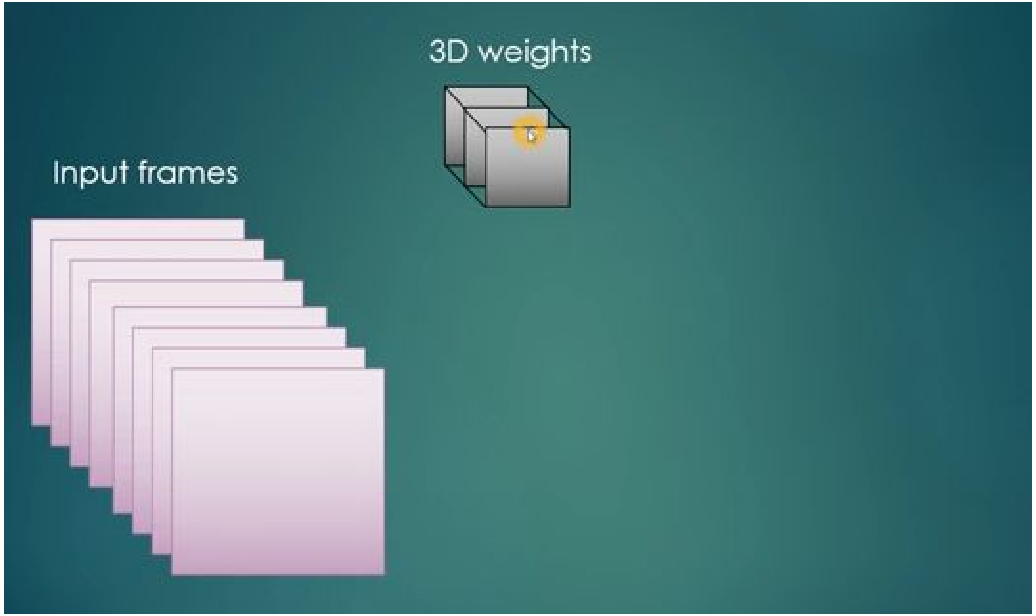




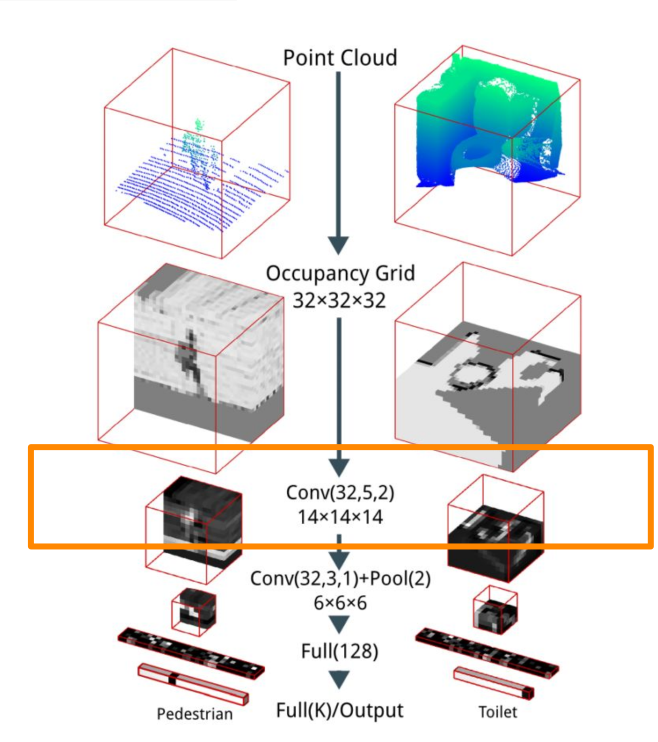
Deep Learning Model: VoxNet (3D CNN 3D objects)



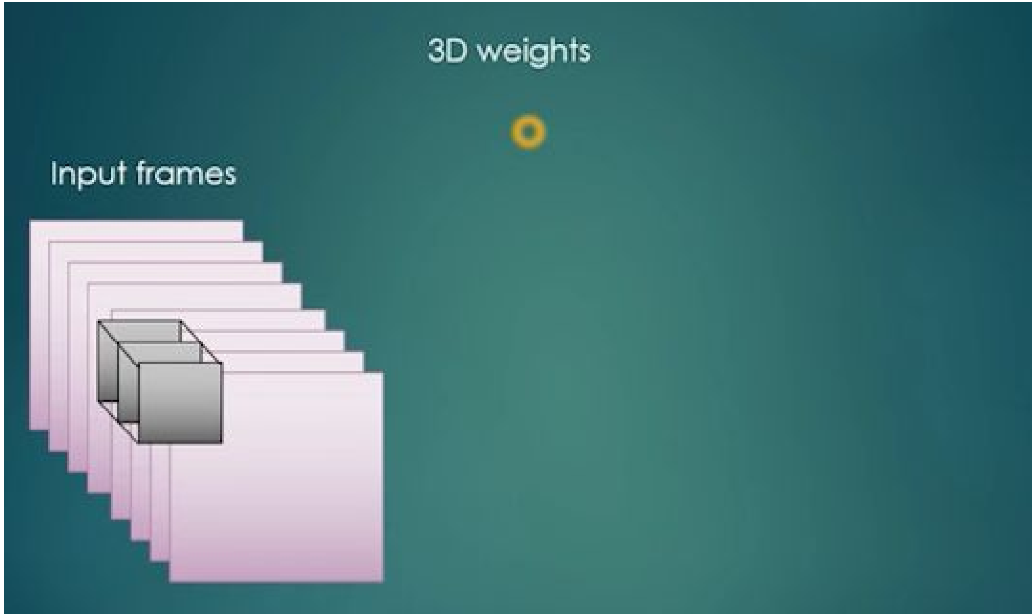
- Convolution 3D



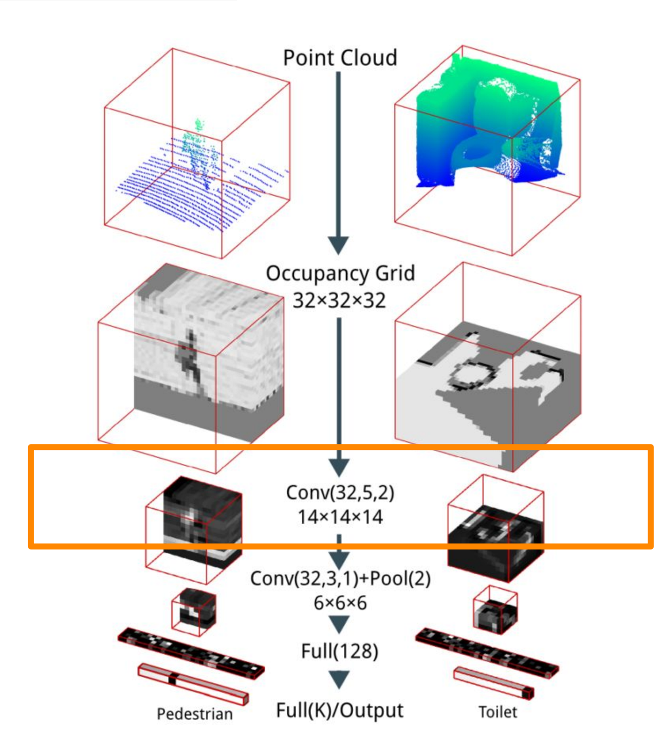
Deep Learning Model: VoxNet (3D CNN 3D objects)



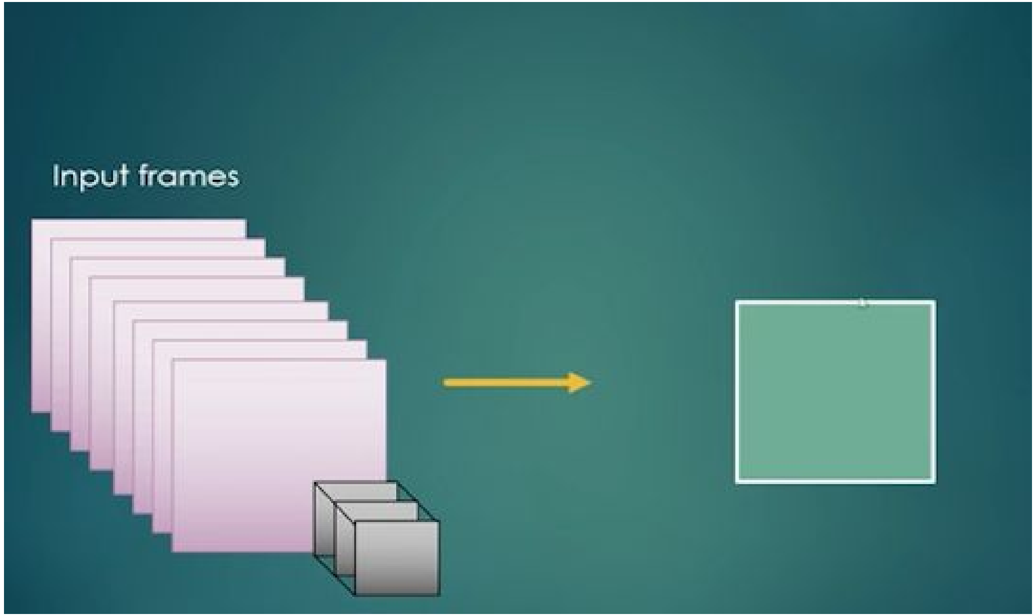
- Convolution 3D



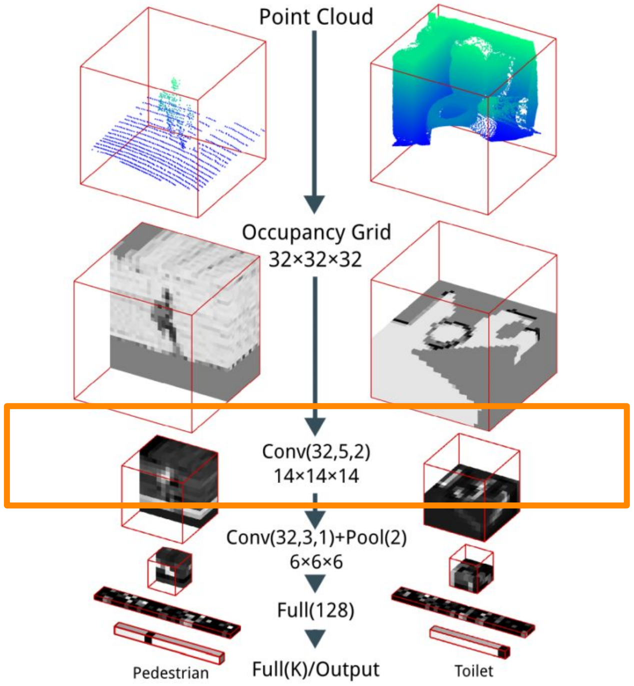
Deep Learning Model: VoxNet (3D CNN 3D objects)



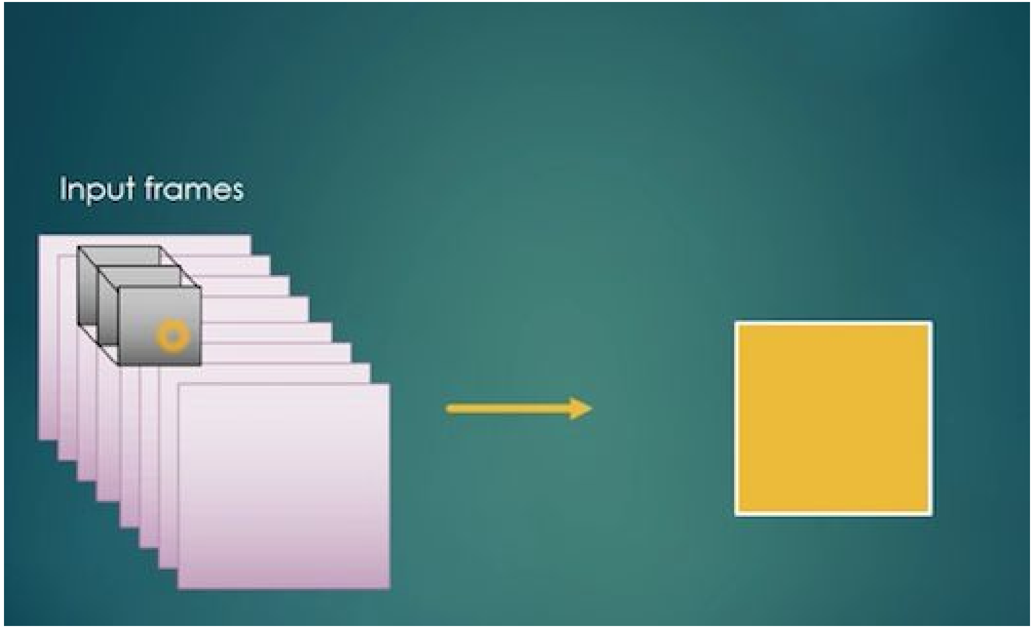
- Convolution 3D



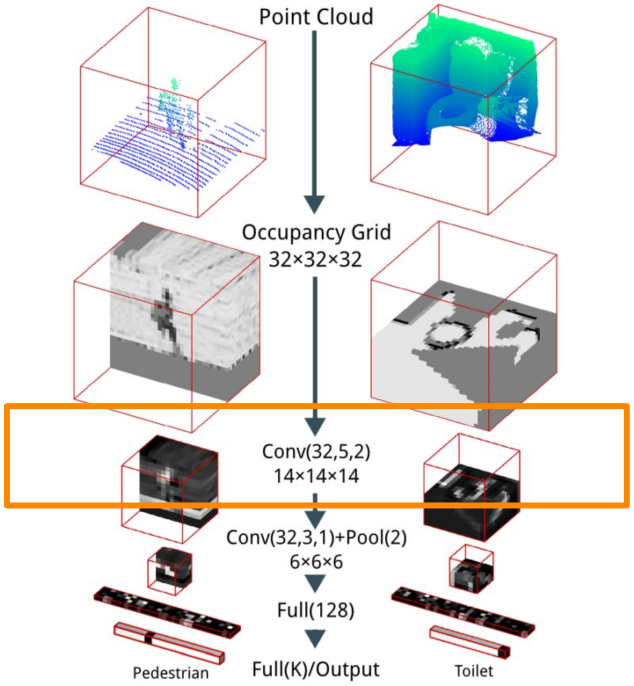
Deep Learning Model: VoxNet (3D CNN 3D objects)



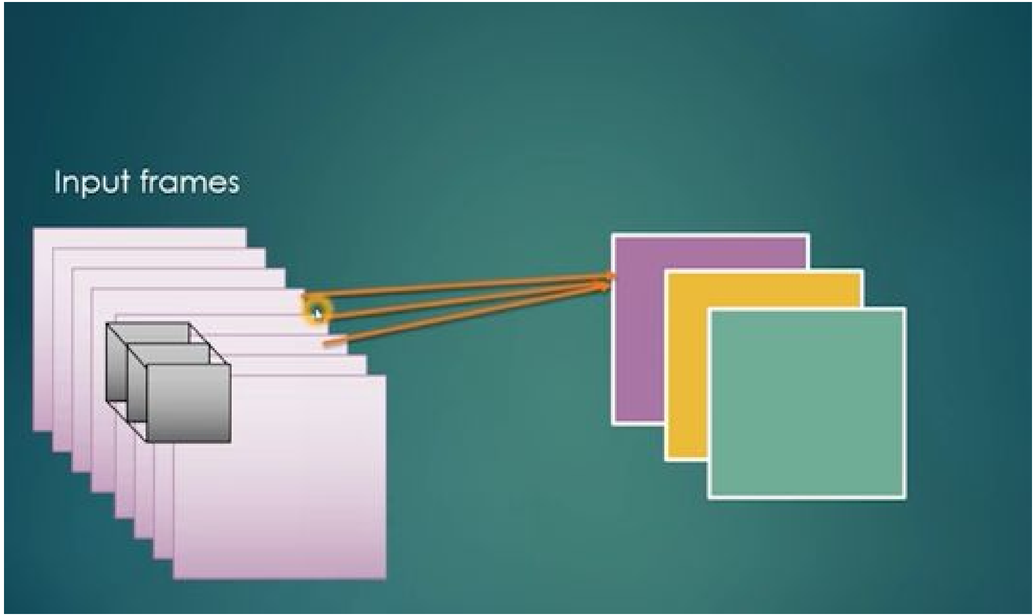
- Convolution 3D



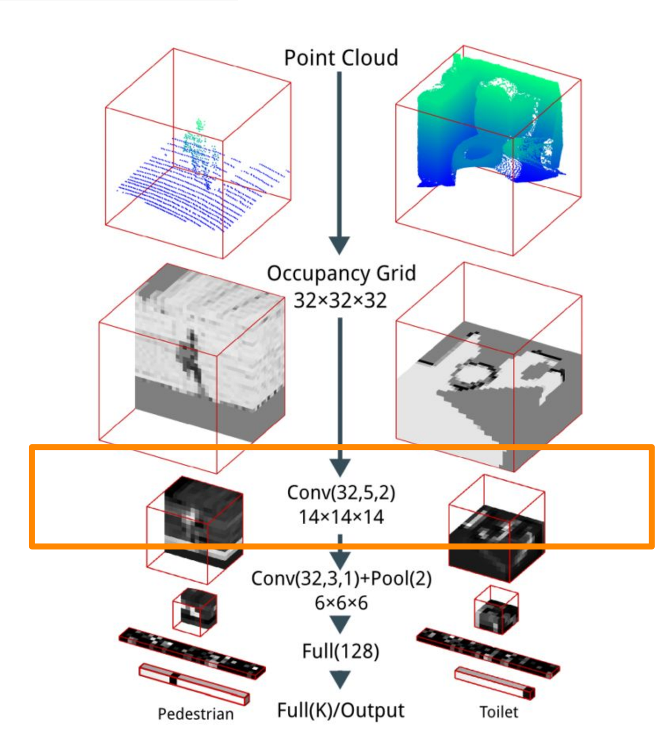
Deep Learning Model: VoxNet (3D CNN 3D objects)



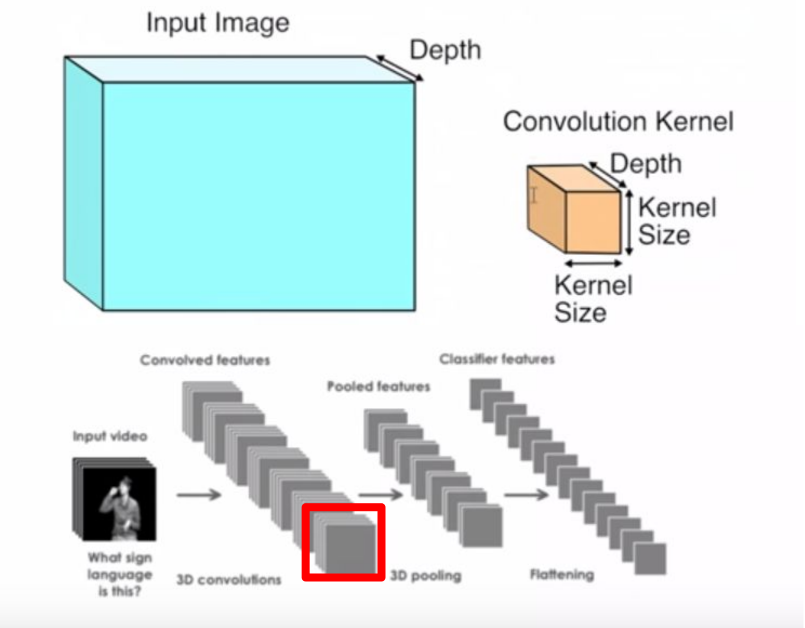
- Convolution 3D



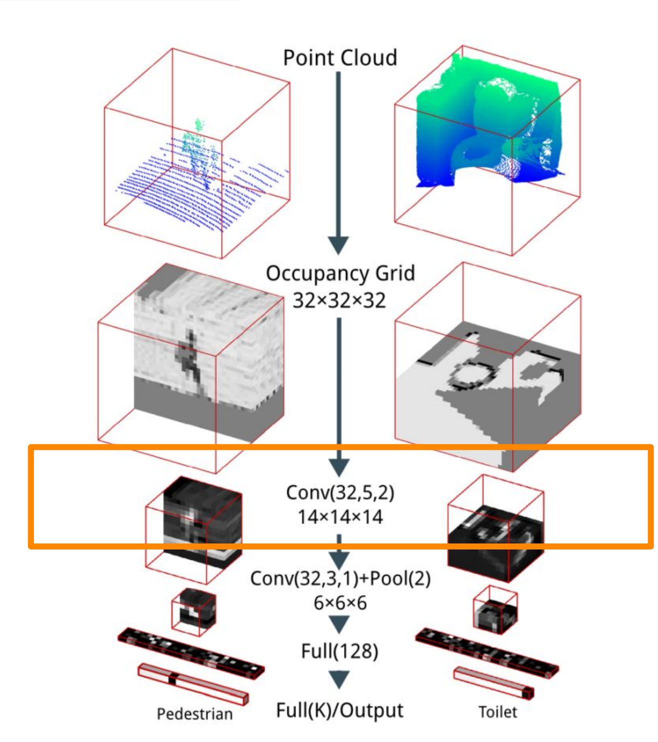
Deep Learning Model: VoxNet (3D CNN 3D objects)



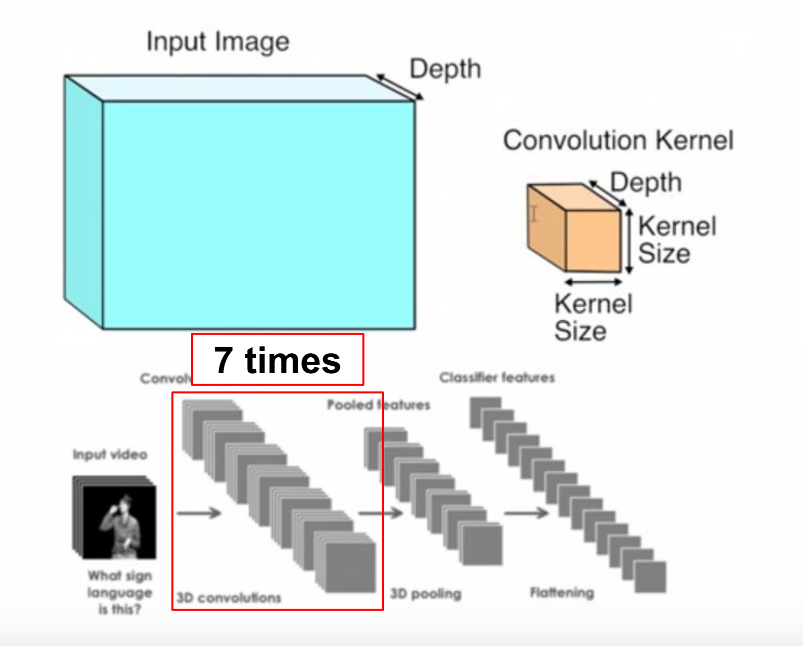
- Convolution 3D



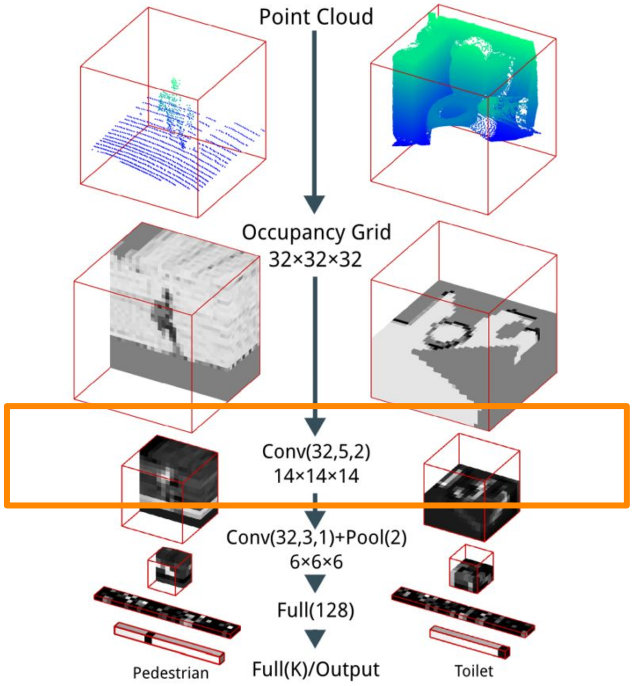
Deep Learning Model: VoxNet (3D CNN 3D objects)



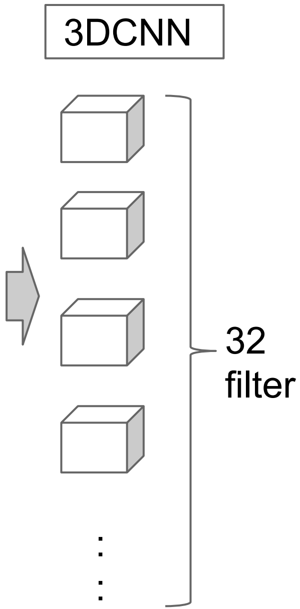
- Convolution 3D



Deep Learning Model: VoxNet (3D CNN 3D objects)



▸ Convolution 3D

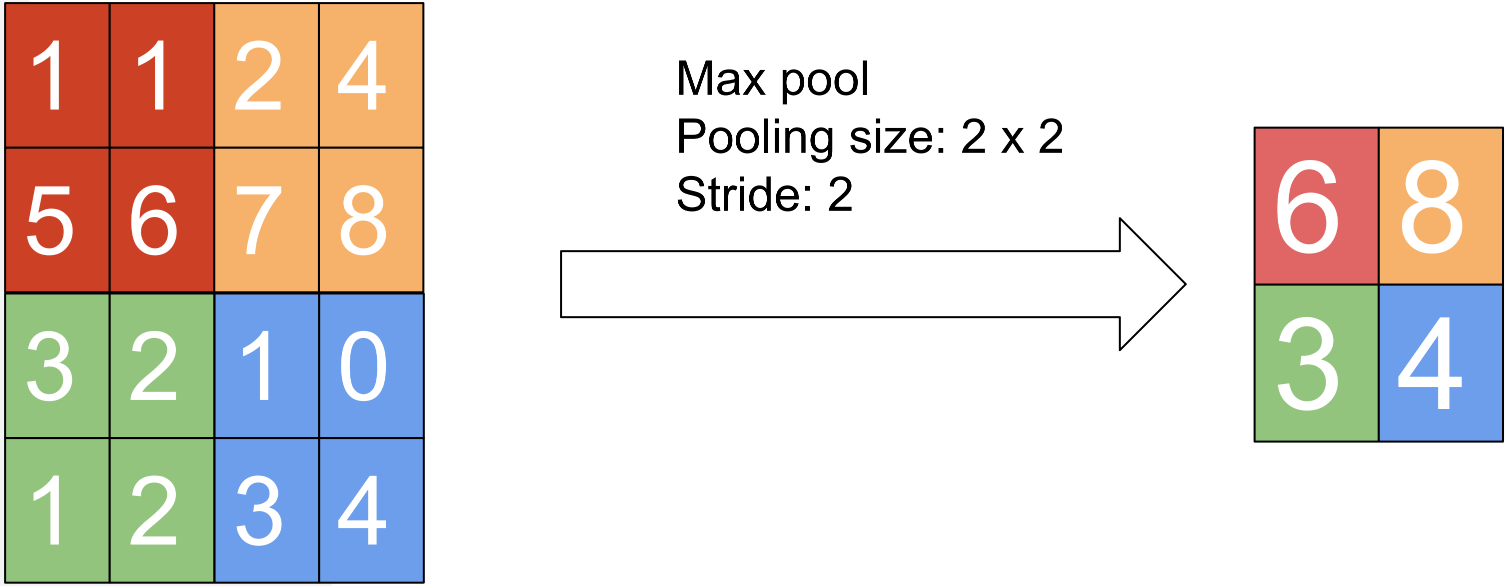



```
Conv3D(input_shape=(32, 32, 32, 1),
       kernel_size=(5, 5, 5),
       strides=(2, 2, 2),
       data_format="channels_last"
)
```

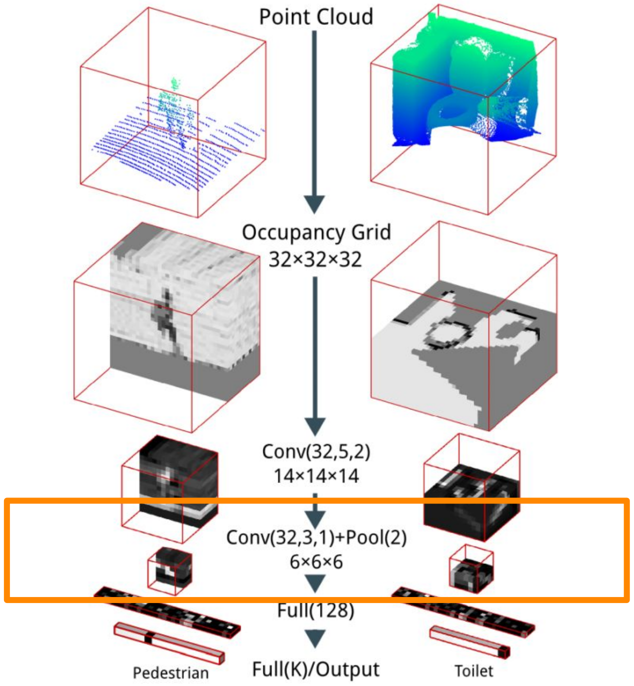
Deep Learning Model: VoxNet (Max Pool3D)

- ## Max Pool 2D

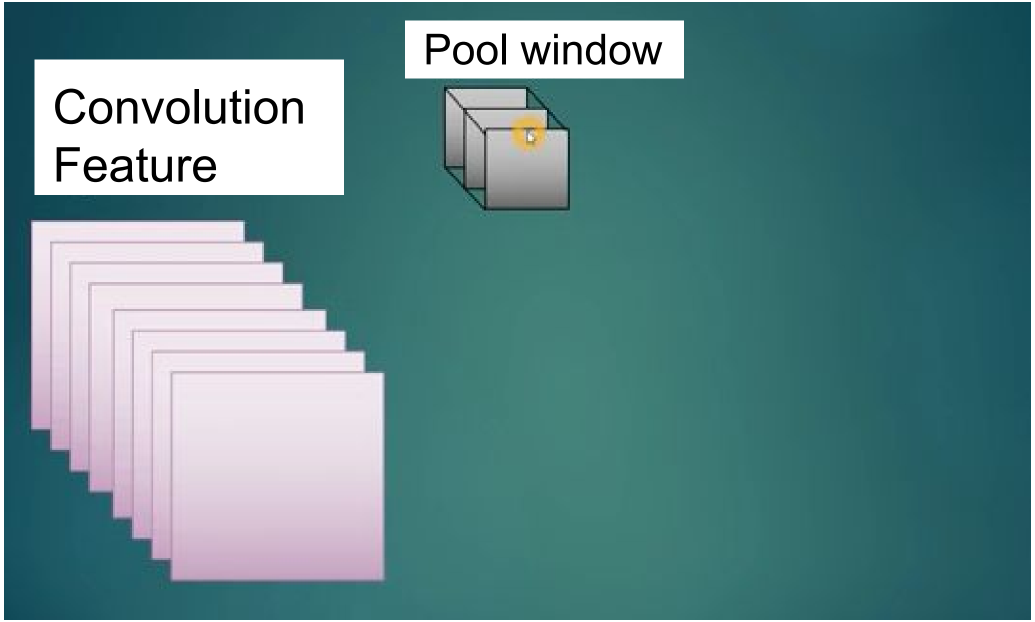
Convolution Feature



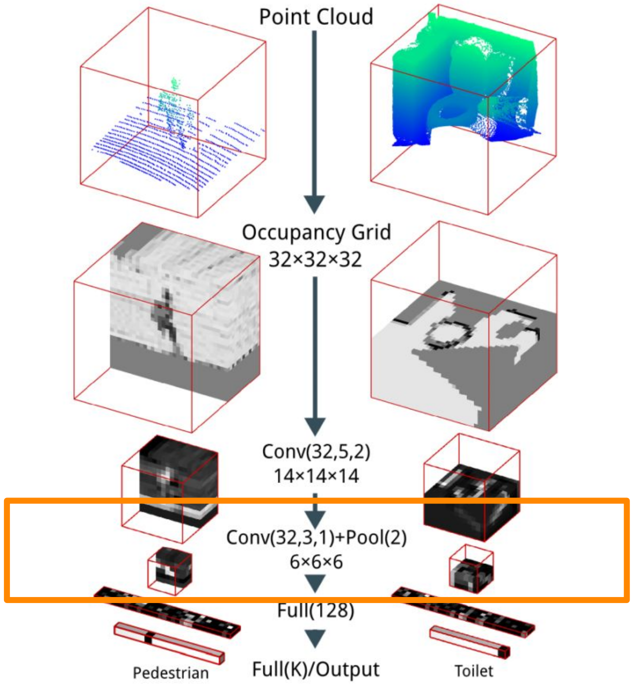
Deep Learning Model: VoxNet (Max Pool3D)



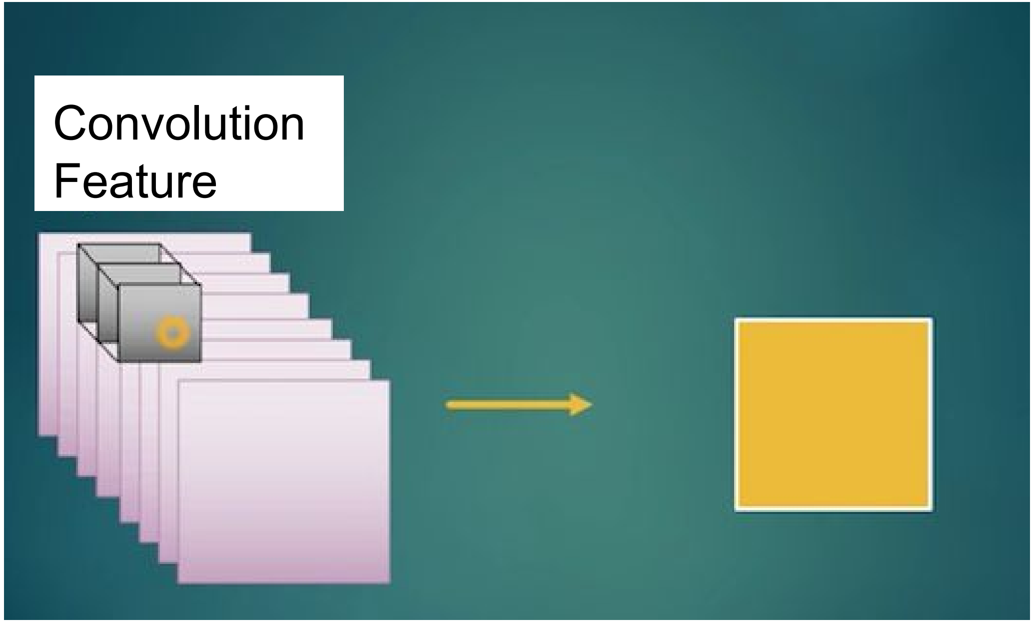
- Max Pool 3D



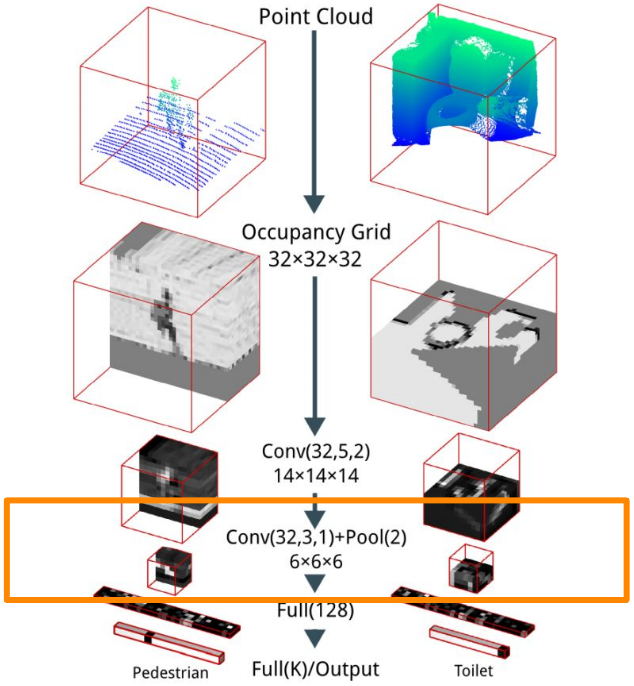
Deep Learning Model: VoxNet (Max Pool3D)



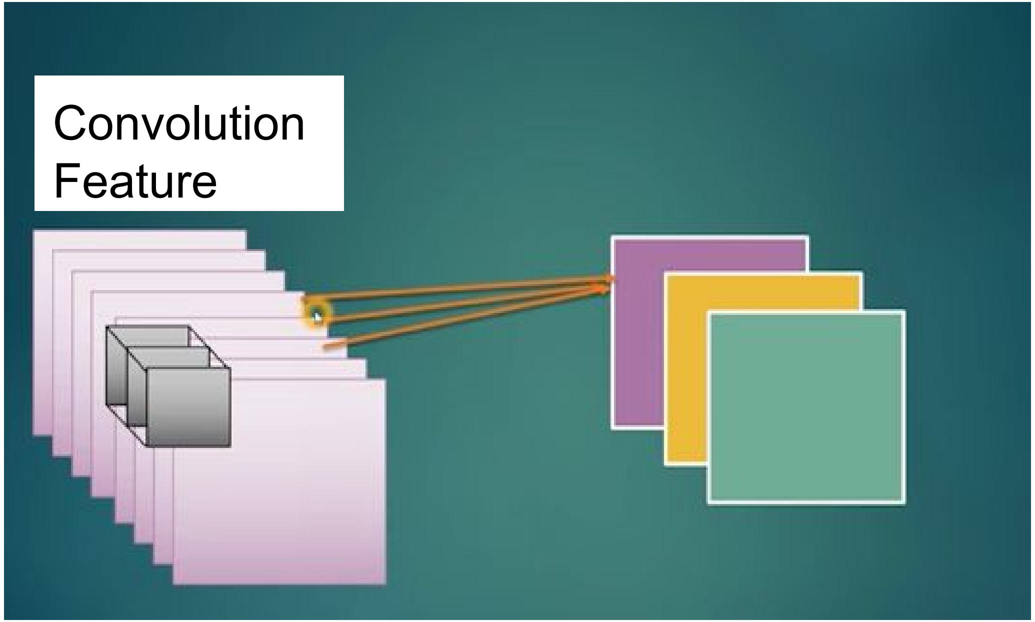
- Max Pool 3D



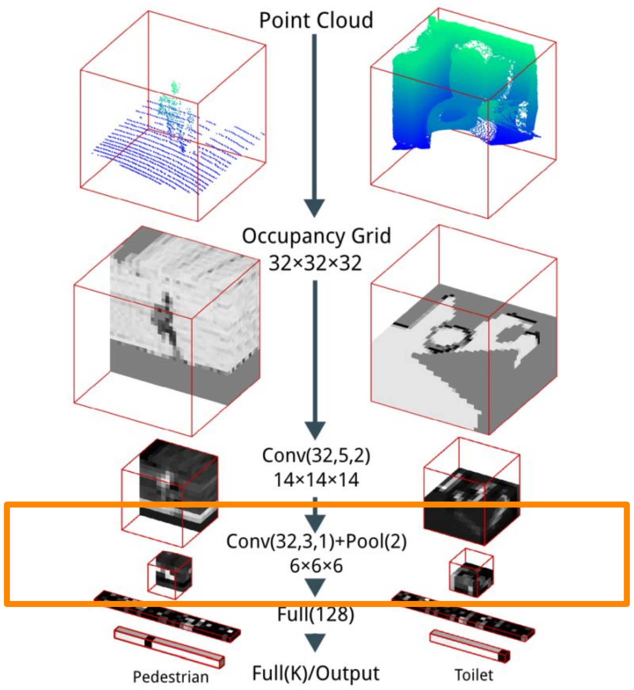
Deep Learning Model: VoxNet (Max Pool3D)



- Max Pool 3D



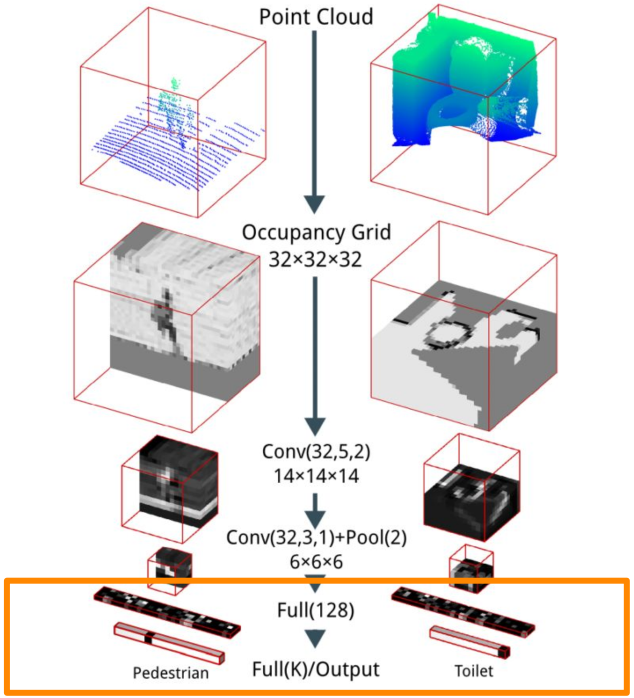
Deep Learning Model: VoxNet (Max Pool3D)



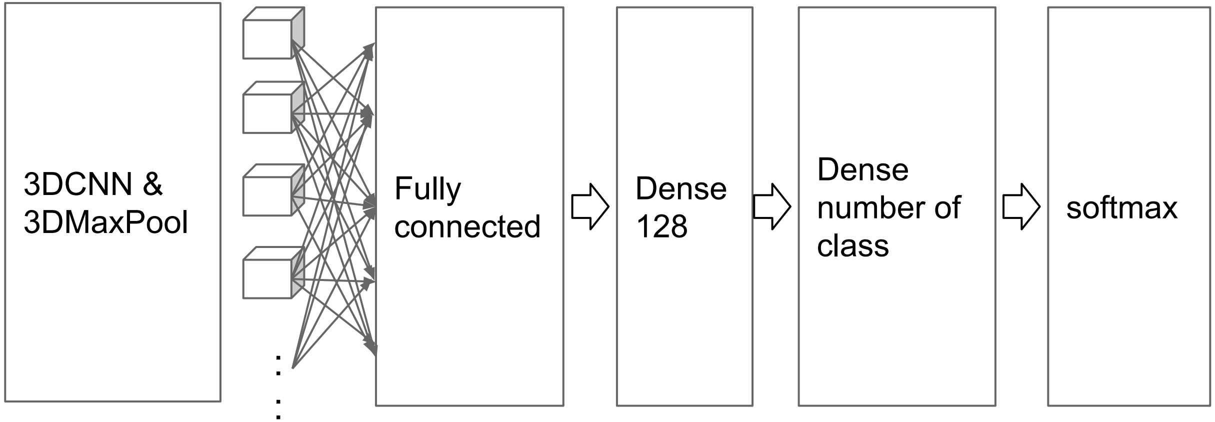
- Max Pool 3D

```
MaxPooling3D(pool_size=(2, 2, 2),
             data_format='channels_last',)
```

Deep Learning Model: VoxNet (Max Pool3D)



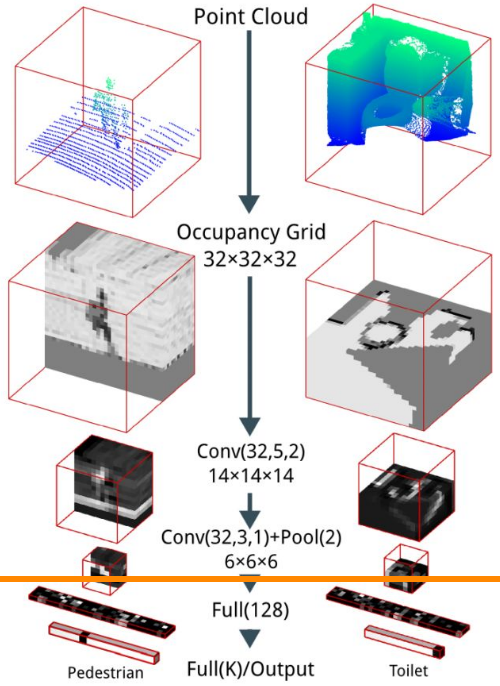
- Fully Connected and Output



Deep Learning Model: VoxNet (Flatten, Dense)

- Fully Connected and Output
  - Softmax function
    - It maps output as a probability distribution
    - It is easy to differentiate

Deep Learning Model: VoxNet (Output)

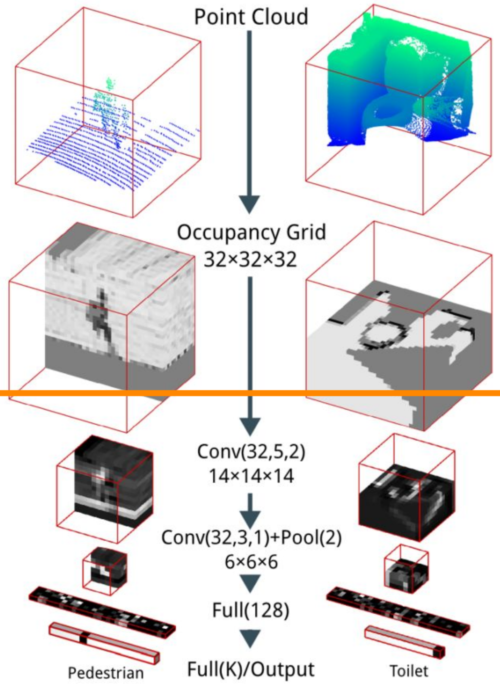


- Fully Connected and Output

```
model.add(Flatten())
model.add(Dense(128, activation='linear',))
model.add(Dense(output_dim=number_class,
                      activation='linear',))

model.add(Activation("softmax"))
```

Deep Learning Model: VoxNet



▸ Model

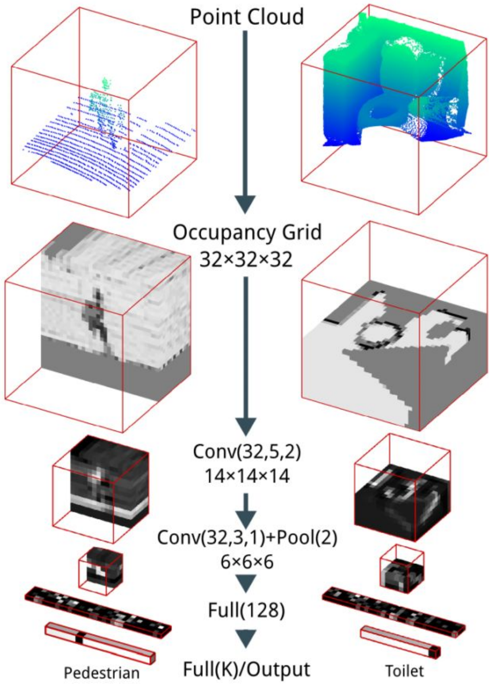
```
model = Sequential()
model.add(Conv3D(input_shape=(32, 32, 32, 1),
                 kernel_size=(5, 5, 5), strides=(2, 2, 2),
                 data_format='channels_last',))
model.add(Conv3D(input_shape=(32, 32, 32, 1),
                 kernel_size=(3, 3, 3), strides=(1, 1, 1),
                 data_format='channels_last',))

model.add(MaxPooling3D(pool_size=(2, 2, 2),
                       data_format='channels_last',))

model.add(Flatten())
model.add(Dense(128, activation='linear',))

model.add(Dense(output_dim=number_class, activation='linear',))
model.add(Activation('softmax'))
```

Deep Learning Model: VoxNet



- Train

```
model.compile(loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_voxel_data, y_class_label)
```

Specific Case

Improve technique accuracy

Improve technique

- Improving accuracy has 2 approaches
  - Model
    - Advantage
      - A variety of ways to improve accuracy
    - Disadvantage
      - A deep model takes a lot of resources
      - It is not obvious which model is better
  - Data
    - Advantage
      - The effect of changes are obvious
    - Disadvantage
      - Approaches are limited

Improve accuracy

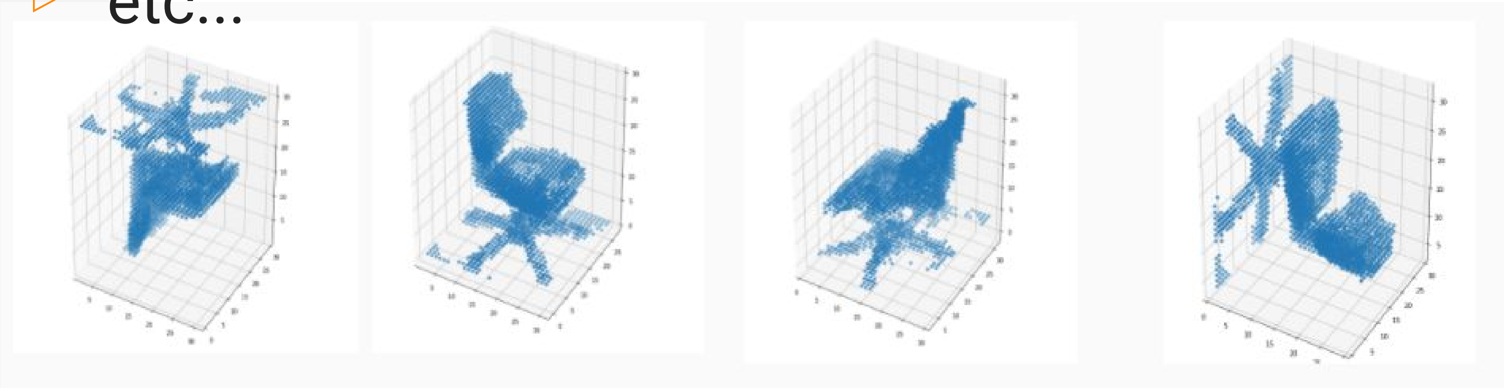
- Improve accuracy for validation data(In my case)
  - Model
    - RandomDropout
    - LeakyRelu
  - Data
    - Data augmentation(3D data)
      - Data increase
    - Class weight for Unbalanced category data

Improve technique

- Data Approach
  - Data Augmentation has advantages over other methods
    - The effects are obvious
    - It does not increase calculation time unlike adding layers to the model

Improve technique

- Data Augmentation
  - Rotation
  - Shift
  - Shear
  - etc...



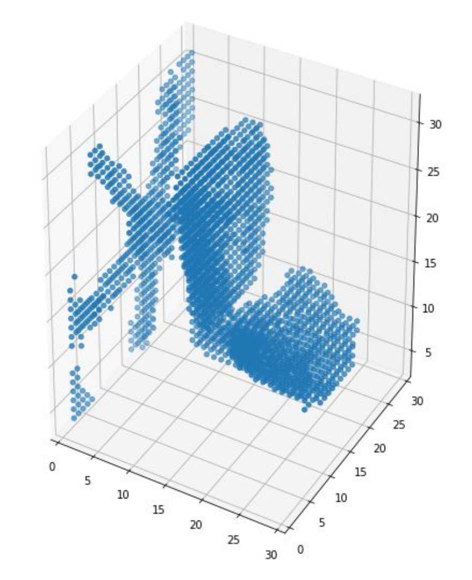
Improve technique

- Data Augmentation 3D
  - Augmentation_matrix is changing

```
channel_images = [ndi.interpolation.affine_transform(x,
                                                     augmentation_matrix,
                                                    )for x in x_voxel]
x = np.stack(channel_images, axis=0)
```

Improve technique

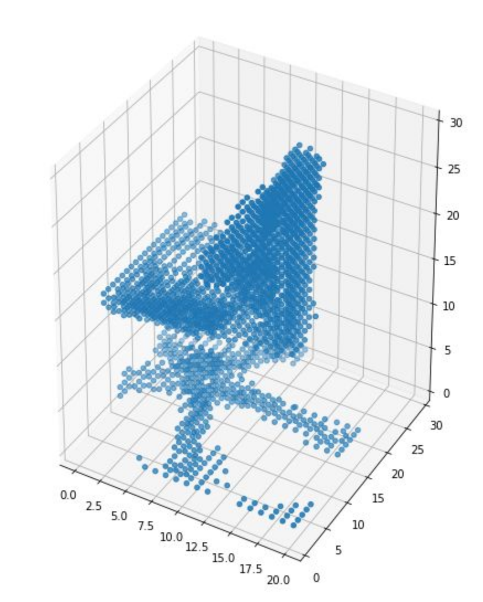
Data Augmentation 3D Rotation




```
rotation_matrix_y = np.array([[np.cos(theta), 0, np.sin(theta) , 0],
                              [0            , 1, 0             , 0],
                              [-np.sin(theta), 0, np.cos(theta), 0],
                              [0            , 0 , 0            , 1]])
```

Improve technique

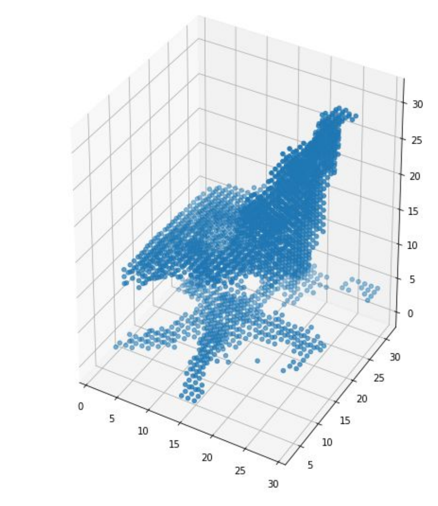
Data Augmentation 3D Shift




```
shift_matrix = np.array([[1, 0, 0, shift_x],
                         [0, 1, 0, shift_y],
                         [0, 0, 1, shift_z],
                         [0, 0, 0, 1      ]])
```

Improve technique

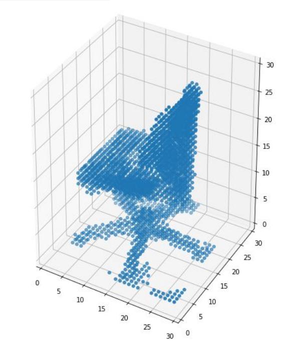
Data Augmentation 3D Shear



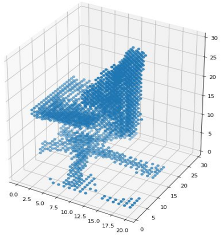

```
shear_matrix = np.array([[1       , shear_x, shear_x, 0],
                         [shear_y, 1       , shear_y, 0],
                         [shear_z, shear_z, 1       , 0],
                         [0      , 0       , 0       , 1]
                         ])
```

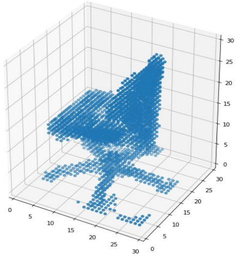
Improve accuracy

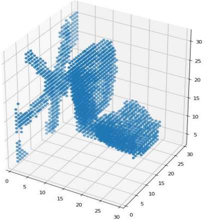
- Data increase



Add Data augmented data to Training data







Specific Case

Improve technique speed

Improve calculation speed

- Deep Learning has a lot of ways to improve calculation speed (In my case)
  - Use a GPU(GeForce GTX 1080: Memory 8GB)
  - CPU optimization
  - Multi thread
  - Prepare feature set

Improve calculation speed

- CPU optimize (TensorFlow build option)

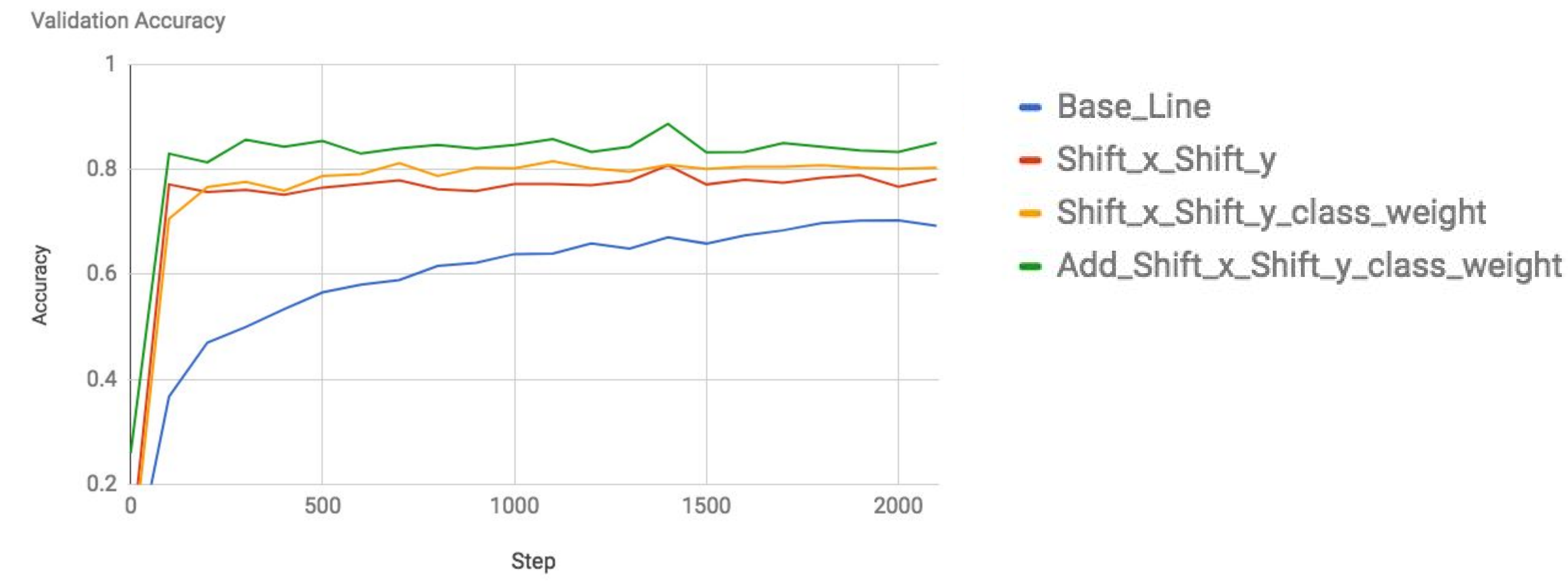
```
bazel build -c opt --copt=-mavx --copt=-mavx2 --copt=-mfma
```

Specific Case

Results

Results

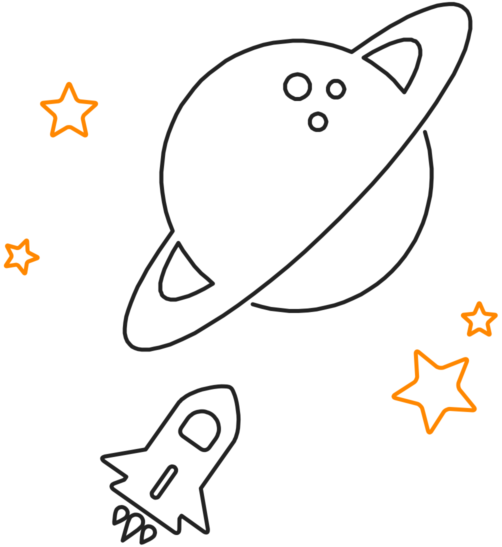
Validation Accuracy



Result

Method	Explanation	Training (accuracy)	Validation (accuracy)
BaseLine	BaseLine	90%	79%
Shift_x_Shift_y	Data augmentation(x-shift, y-shift)	80%	80%
Shift_x_Shift_y_class_weight	Data augmentation(x-shift, y-shift) + class weight	80%	83%
Add_Shift_x_Shift_y_class_weight	Data augmentation(x-shift, y-shift) + class weight + ADD(x-shift, y-shift)	85%	**85%**

Conclusion



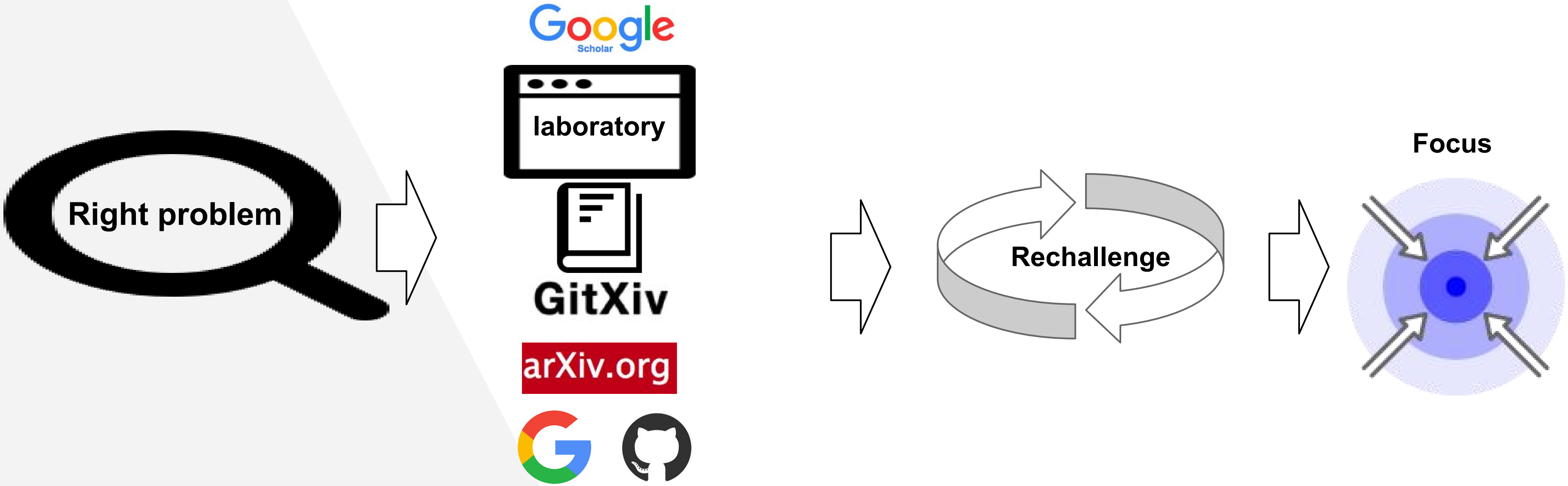
Conclusion

Summary of this presentation

Conclusion

- Strategy

Right Problem | **Right Method** | **Rechallenge** | **Focus**



Ref: https://www.iconfinder.com/, https://github.com/, https://arxiv.org/, http://www.gitxiv.com/, https://scholar.google.co.jp/

Conclusion

Our case

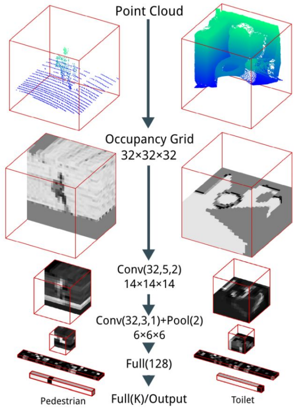
Right Problem

3D object recognition

On-demand manufacturing service

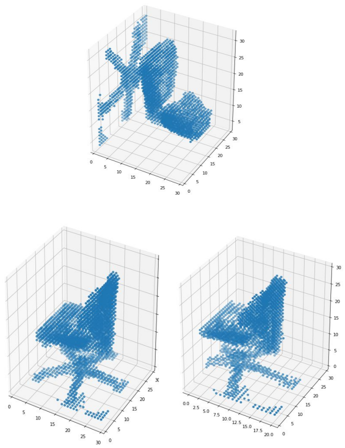
Right Method

VoxNet

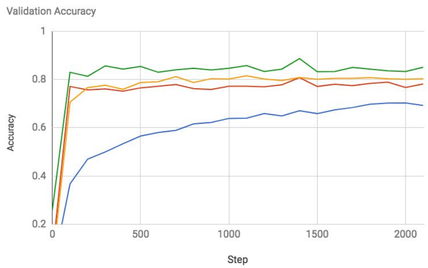


Rechallenge

Data augmentation
Customize model



Focus



We're hiring

Recruiting

Deep Learning For 3D objects

It is a rare case, Implementing deep learning for 3D objects

We are hiring!!

- Deep Learning for 3D objects

- Working in Japan

https://www.wantedly.com/projects/111707

contact@kabuku.co.jp

THANKS!

Any questions?

You can find me at @SnowGushiGit & masaya.ohgushi@kabuku.co.jp

References

References

- *MNIST datasets*
  - https://www.tensorflow.org/get_started/mnist/beginners
- *Flaticon*
  - [*www.flaticon.com*](http://www.flaticon.com)
- 3D CNN-Action Recognition Part-1
  - https://www.youtube.com/watch?v=ecbeIRVqD7g&t=82s
- Bengio, Yoshua, et al. "Curriculum learning." *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009.
- He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.
- Yann, Margot Lisa-Jing, and Yichuan Tang. "Learning Deep Convolutional Neural Networks for X-Ray Protein Crystallization Image Analysis." *AAAI*. 2016.
- Maturana, Daniel, and Sebastian Scherer. "Voxnet: A 3d convolutional neural network for real-time object recognition." *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015.

References

- Deep Learning Book
  - http://www.deeplearningbook.org/
- IconFinder
  - https://www.iconfinder.com/,
- Github
  - https://github.com/,
- Arxiv
  - https://arxiv.org/
- GitXiv
  - http://www.gitxiv.com/
- GoogleSchlor
  - https://scholar.google.co.jp/