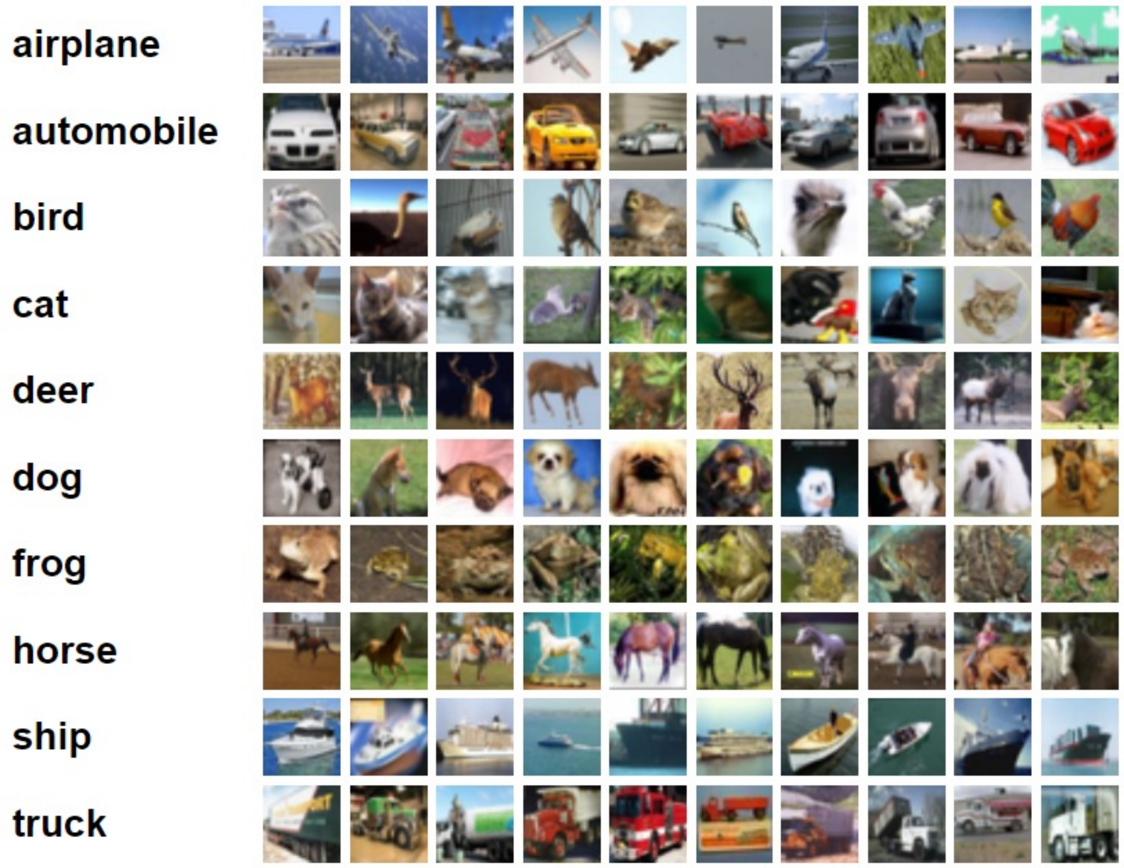
HW2 Instructions

Deep Learning

Tasks

- Design an CNN architecture
- Implement batch normalization
- Implement pooling layer
- Implement residual block
- Implement fully convolutional residual network **without fully connected layers or MLP**

Dataset: CIFAR-10



- 32 x 32 color images
- 10 classes, with 6000 images per class
- Training set: 50,000 images
- Test set: 10,000 images
- The classes are completely mutually exclusive

More information: https://www.cs.toronto.edu/~kriz/cifar.html

Grading

- Part 4: Batch Normalization 20%
- Part 6: Residual Block 20%
- Part 7: Residual Network 20%
- Model size 15%:
  - 10%: If your model is smaller than **6MB**, you will get 10%.
  - 5%: The remaining 5% will depend on your ranking within the class.

Grading

- Model accuracy 15%:
  - 10%: If your accuracy is higher than **78%**, you will get 10%.
  - 5%: The remaining 5% will depend on your ranking within the class.
- Model accuracy on another dataset 10%: it will depend on your ranking within the class.

Rules

- Please do NOT call any other existing libraries for implementations.
- You can use tensor functions in torch.
- You can use nn.Sequential(), nn.Identity() and nn.ModuleList() to implement your residual block in part 6.
- You can only use nn.Flatten() to flatten your tensors in part 7.
- Don’t directly call torch.nn functions for other implementations.

Part 4: Batch normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

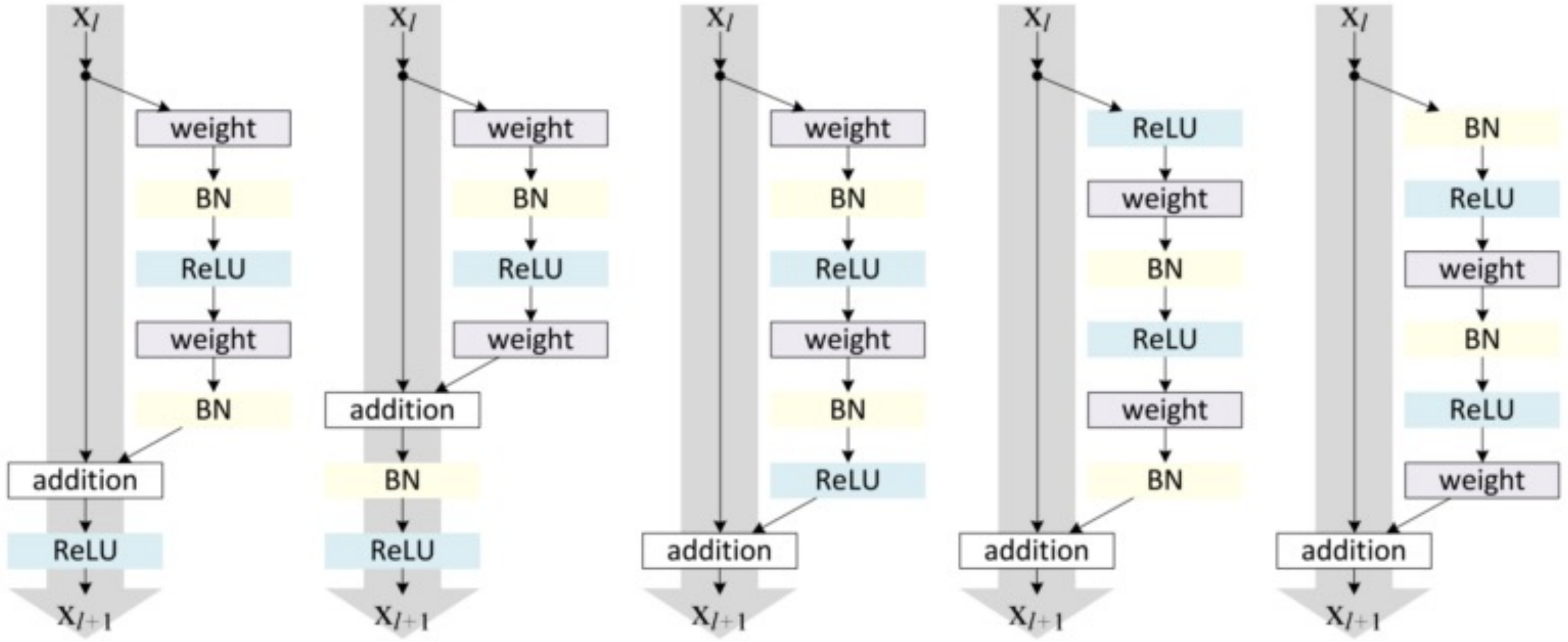
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Part 4: Batch normalization

```
def batch_norm(self, x, gamma, beta, moving_mean, moving_var, eps, momentum):

    if not torch.is_grad_enabled():
        x_hat = (x - moving_mean) / torch.sqrt(moving_var + eps)

    else:
        ################################################
        # Please fill this part by your implementation #
        ################################################

    y = gamma * x_hat + beta

    return y, moving_mean.data, moving_var.data
```

Part 6: Residual blocks



Part 7: CNN architecture

- myBatchNorm (Part 4)
- myConvolution (Part 5)
- myActivation (Part 5)
- myMaxPooling (Part 5)
- myAvgPooling (Part 5)
- myResBlock (Part 6)
- Part 5: DO NOT MODIFY
- **no fully connected layers or MLP**

```
# example: 5 convolutional layer CNN + 2 hidden layer MLP
self.cnn = nn.Sequential(
    # CNN
    myConvolution(input_channel, 256, 3, 1, 1),
    myActivation(),
    myMaxPooling(2),
    myConvolution(256, 256, 3, 1, 1),
    myActivation(),
    myMaxPooling(2),
    myConvolution(256, 256, 3, 1, 1),
    myActivation(),
    myMaxPooling(2),
    myConvolution(256, 256, 3, 1, 1),
    myActivation(),
    myMaxPooling(2),
    myConvolution(256, 256, 3, 1, 1),
    myActivation(),
    # MLP
    nn.Flatten(1),
    nn.Linear(1024, 256),
    myActivation(),
    nn.Linear(256, 256),
    myActivation(),
    nn.Linear(256, num_classes)
)
```

Submission

- Upload your zip file to NTU Cool
- **StudentID_HW2.zip**
    - DL_HW2_StudentID.ipynb
    - StudentID_submission.pt
    - StudentID_submission.csv
- **Deadline: 4/8 23:59**