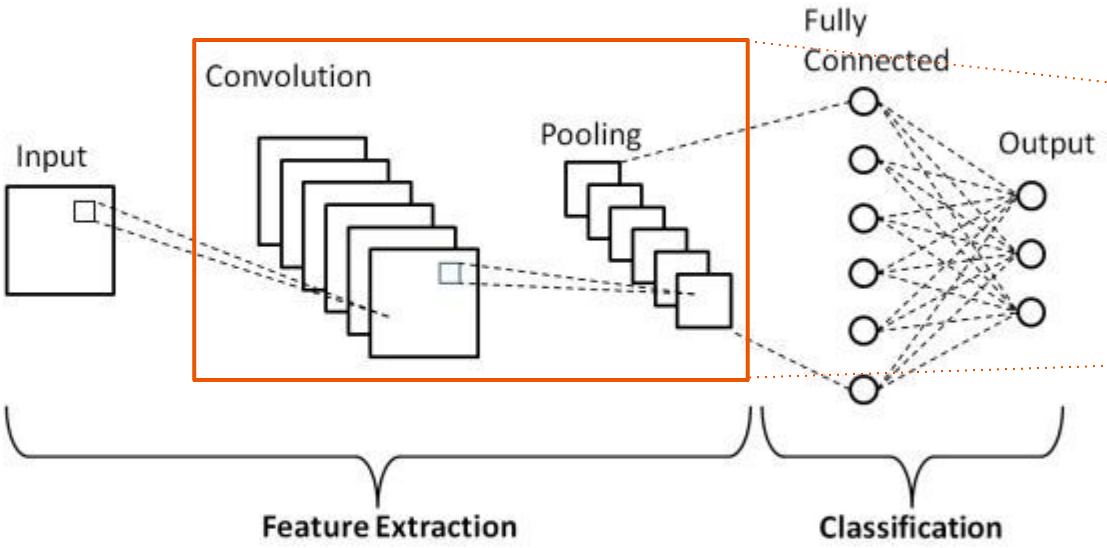
Contrastive Learning

Weiguang Guan, guanw@sharcnet.ca

SHARCNet/Digital Research Alliance of Canada

SHARCNET™ Digital Research Alliance of Canada | Alliance de recherche numérique du Canada

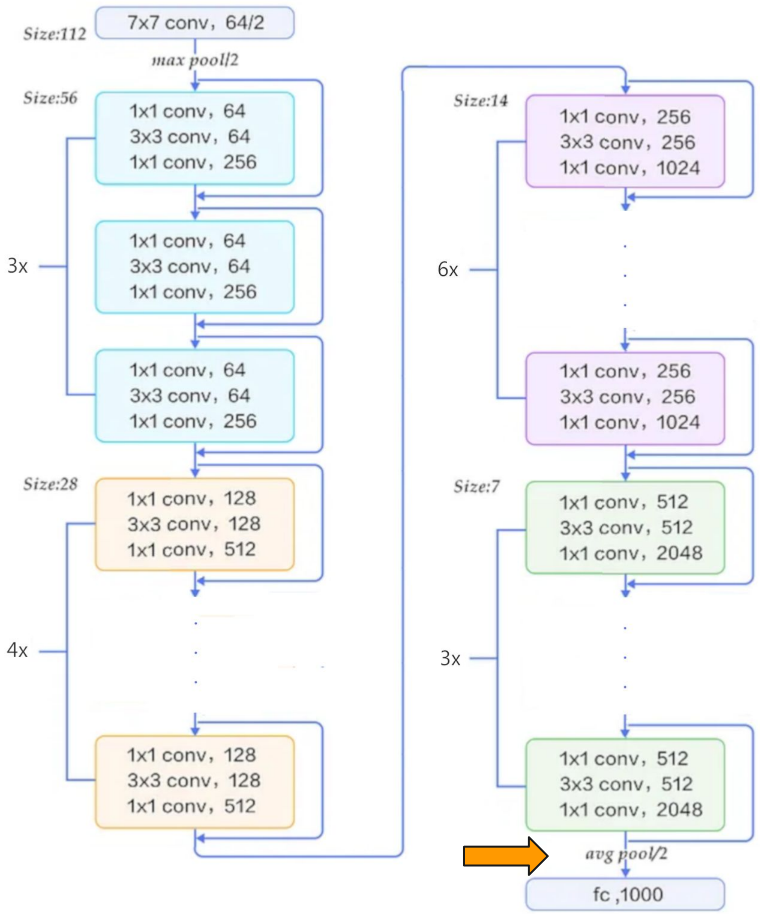
Typical architecture of CNN for image classification



Convolution
Activation
Pooling
BatchNormalization
Dropout

Resnet-50

- A sequences of blocks of convolution layers
- Last (head) layer is a fully connected layer
- The input to the FC layer (or the output of the global average pooling) is a vector of length 2048, which is the final feature used for classification



Today's task

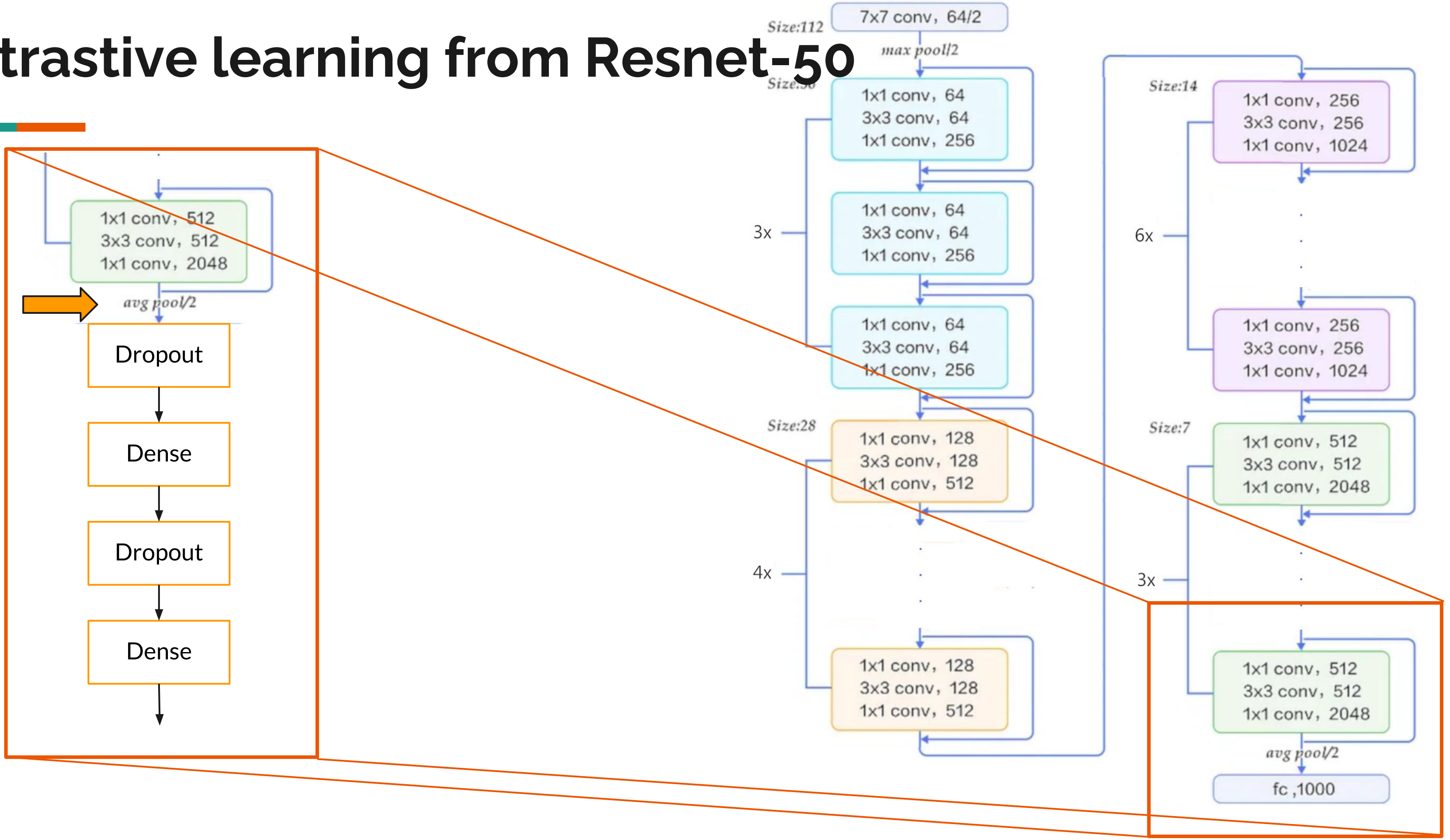
Task: Train the Resnet-50 model to classify Cifar-10 dataset (32×32×3)

Source: https://keras.io/examples/vision/supervised-contrastive-learning/

Comparison:

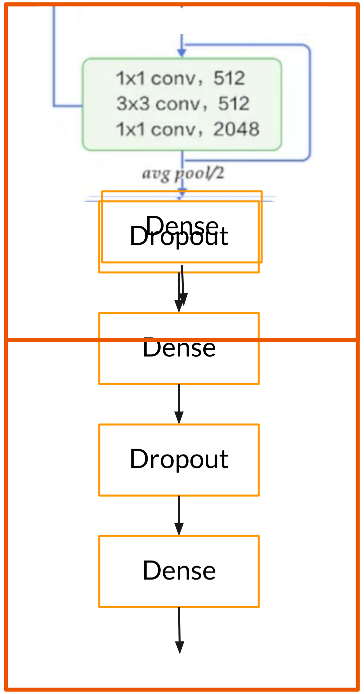
- Train the NN in the traditional way
- Contrastive training

Contrastive learning from Resnet-50



Contrastive learning

- Train the base model with an added top layer so that its output (feature in the form of vector)
  - maximizes the difference between samples of different classes
  - minimizes the difference between samples of the same class
- Then train the entire NN with base model frozen (only train the added layers)

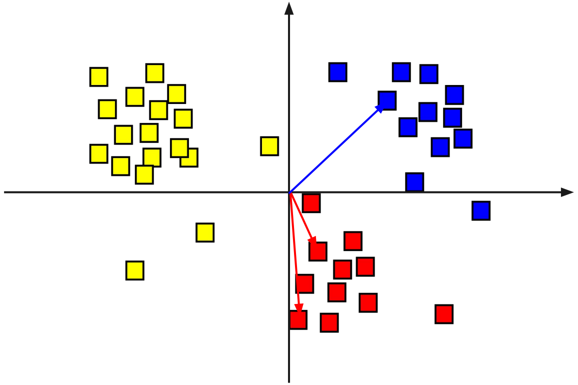


Similarity measurement between features

Suppose v_1 and v_2 are two (row) feature vectors, then

$$\text{Similarity}(v_1, v_2) = (v_1/|v_1|)\,(v_2^T/|v_2|) = \cos(\theta),$$

where θ is the angle between v_1 and v_2



tfa.losses.npairs_loss(y_true, y_pred)

- **y_true**: [batch_size]

  Labels of samples

- **y_pred**: [batch_size, batch_size]

  Element [i, j] of the matrix represents the similarity of sample i with sample j

tfa.losses.npairs_loss(y_true, y_pred) cont.

Suppose a mini-batch has labels y_true and v_i is a row vector associated with sample i, then

- Calculate similarity matrix (y_pred) before calling this loss function

$$y\_pred[i,j] = v_i \cdot v_j^T$$

$$y\_pred = V V^T \text{ (where } V = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \dots \end{bmatrix} \text{)}$$

tfa.losses.npairs_loss(y_true, y_pred) cont.

- Remapping labels (y_true) into matrix of the same shape as y_pred

  If y_true = [1, 8, 5, 1, 7], then

  remapped = equal(y_true, y_true^T) =

```
[ True, False, False,  True, False]
[False,  True, False, False, False]
[False, False,  True, False, False]
[ True, False, False,  True, False]
[False, False, False, False,  True]
```

tfa.losses.npairs_loss(y_true, y_pred) cont.

- Computes softmax cross entropy

  ```
  entropy = nn.softmax_cross_entropy_with_logits(
                  logits=y_pred,
                  labels=remapped
          )
  ```

- loss = reduce_mean(entropy)

Some considerations

- Fine tune after contrastive learning
- Resnet-50 does not show its full power for images smaller than 224x224.
- Better result could be obtained if contrastive learning is combined with transfer learning