CAP6411 Class 8/24

Residual Network (ResNet)

- Before ResNet:
  - Adding more layer to deep CNN results in degradation of performance
  - Is this degradation due to overfitting?
    - No …
    - Because training error went up – if it has been overfitting, training error would have gone down
  - If not overfitting, should adding more layers cause degradation by itself?
    - No …
    - Imagine a shallow network, you can add more layers that are identity mappings, so the resulting deeper network should minimally have the same performance!!

What does this tell us?

- Trying to learn even an identity mapping as network gets deeper is not trivial
- So, what's the solution?
  - Say $\mathcal{H}(\mathbf{x})$ is the desired mapping
  - We learn $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$ instead
  - The original desired mapping is now $\mathcal{F}(\mathbf{x}) + \mathbf{x}$
  - This means that if we want to learn identity mappings, we just need to learn x = 0, which is easier than learning an identity mapping as mentioned

Residual Block

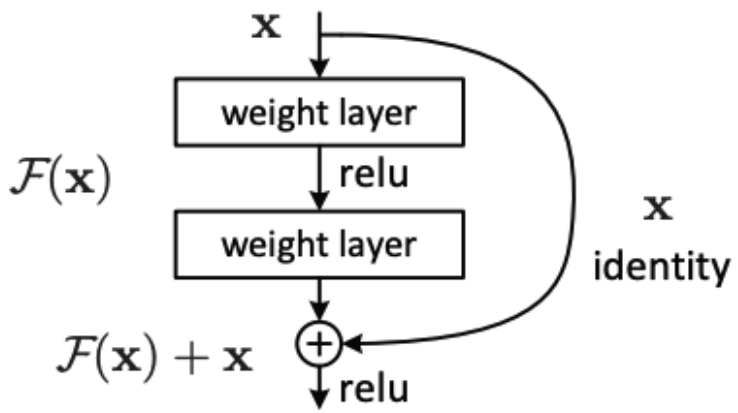


Figure 2. Residual learning: a building block.

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}.$$

$$\mathcal{F} = W_2\sigma(W_1\mathbf{x})$$

- Shortcut connections
- Element-wise addition
- 2-3 layers work best per block

Implementations

3.4. Implementation

Our implementation for ImageNet follows the practice in [21, 41]. The image is resized with its shorter side randomly sampled in $[256, 480]$ for scale augmentation [41]. A 224×224 crop is randomly sampled from an image or its horizontal flip, with the per-pixel mean subtracted [21]. The standard color augmentation in [21] is used. We adopt batch normalization (BN) [16] right after each convolution and before activation, following [16]. We initialize the weights as in [13] and train all plain/residual nets from scratch. We use SGD with a mini-batch size of 256. The learning rate starts from 0.1 and is divided by 10 when the error plateaus, and the models are trained for up to 60×10^4 iterations. We use a weight decay of 0.0001 and a momentum of 0.9. We do not use dropout [14], following the practice in [16].

In testing, for comparison studies we adopt the standard 10-crop testing [21]. For best results, we adopt the fully-convolutional form as in [41, 13], and average the scores at multiple scales (images are resized such that the shorter side is in $\{224, 256, 384, 480, 640\}$).

Assignments (due 8/31)

- Replicate results in Table 2, 3, 4, 5, 6, 7, 8 – 40%
  - No training involved!!
  - Take existing models (many githubs and models on internet) to run INFERENCE to replicate results
- Report and code zip (10%)
  - Errors and obstacles faced running the model
  - Must have conda requirement.txt, cli commands to generate each table above (random checks will be perform to verify)
- Video of live demo (20%)
  - The video should comprise of visual outputs (e.g., a text "cat" overlay on the image being classified that has a cat in it)
- New ideas/insights (30%)