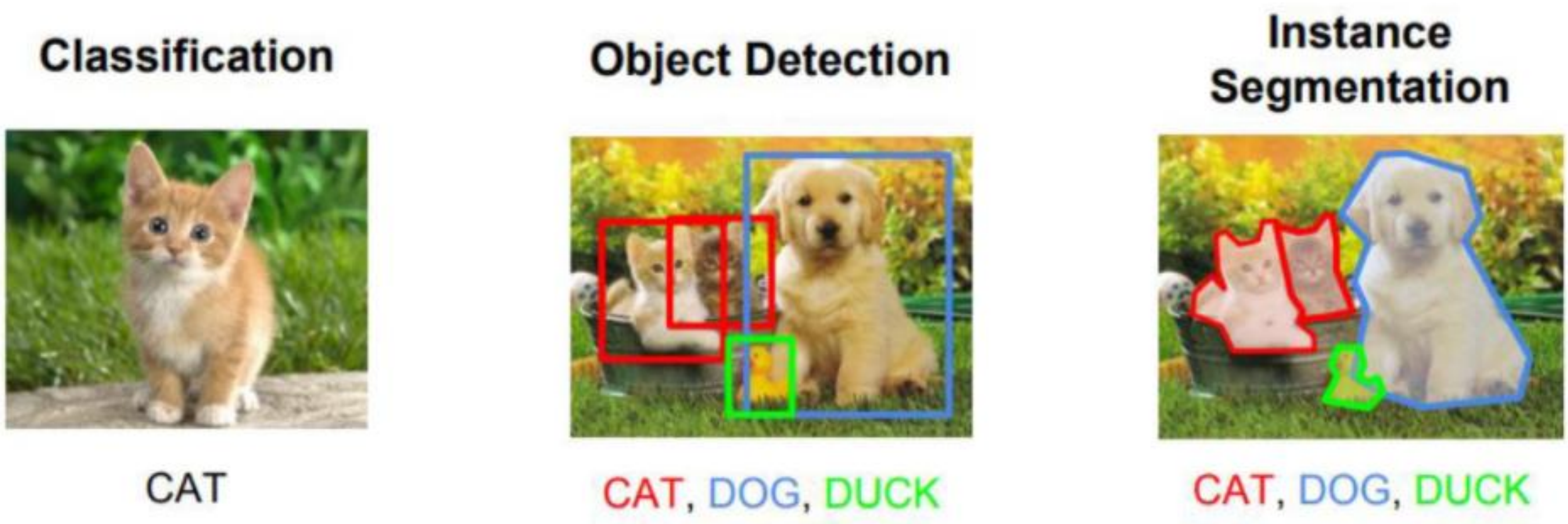
Assignment 1

TA: Jaehoon Lee, Junyong Ahn

Department of Electrical and Computer Engineering
Seoul National University

http://data.snu.ac.kr

Computer vision tasks



Source: https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

- Besides, image reconstruction, image synthesis, style transfer, etc…

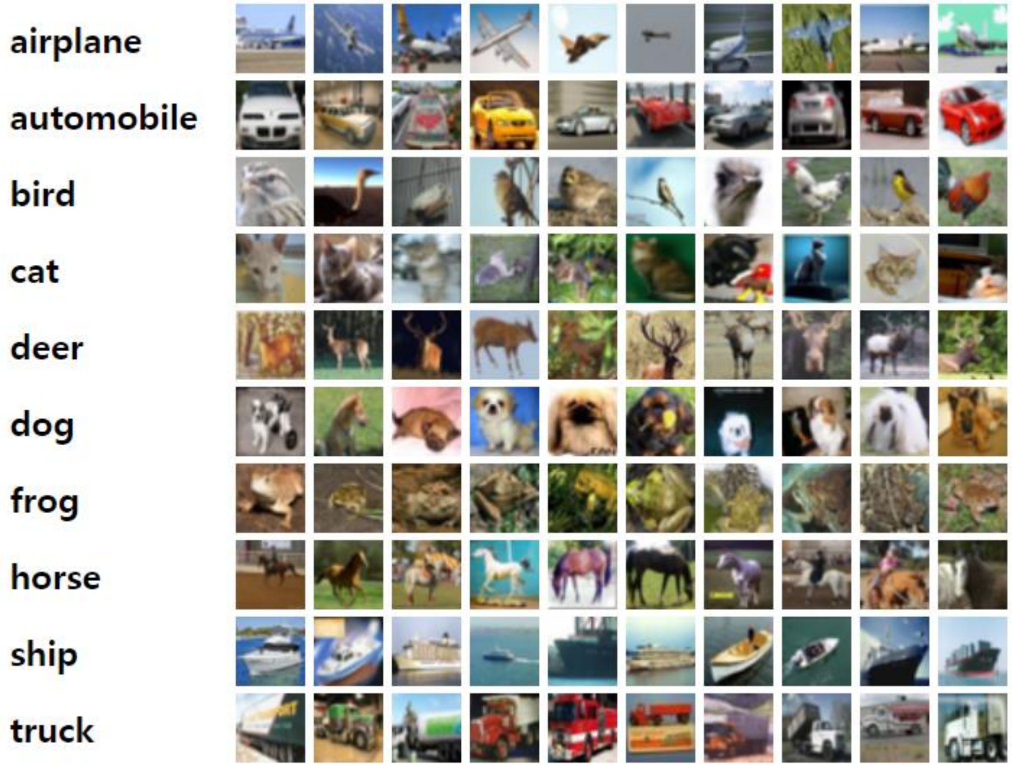
Assignment_1-1 Objective (Convolutional neural network)

- Problem 1: Simple CNN model training on CIFAR-10 dataset
  - 문제에서 제시한 모델 그대로 구현

- Problem 2: Inception module & Residual block 구현
  - 문제에서 제시한 모델 그대로 구현
  - Hyper parameter 변경 가능 (e.g. filter 수)

- Problem 3: Problem 2을 활용한 CNN model training (CIFAR-10)
  - Test set accuracy >= 70%
  - 구현한 모델 설명

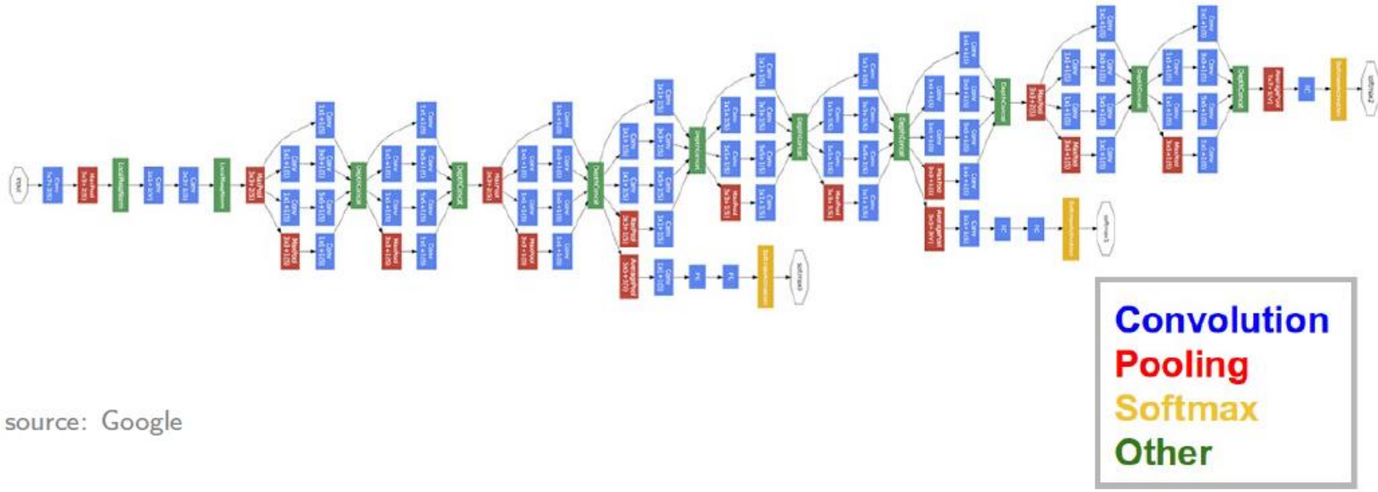
CIFAR-10 dataset



Source: https://www.cs.toronto.edu/~kriz/cifar.htl

- Collection of images used to train machine learning and computer vision algorithms
- Consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class
- There are 50,000 training images and 10,000 test images.
- The classes are completely mutually exclusive (no overlap between truck & automobile).

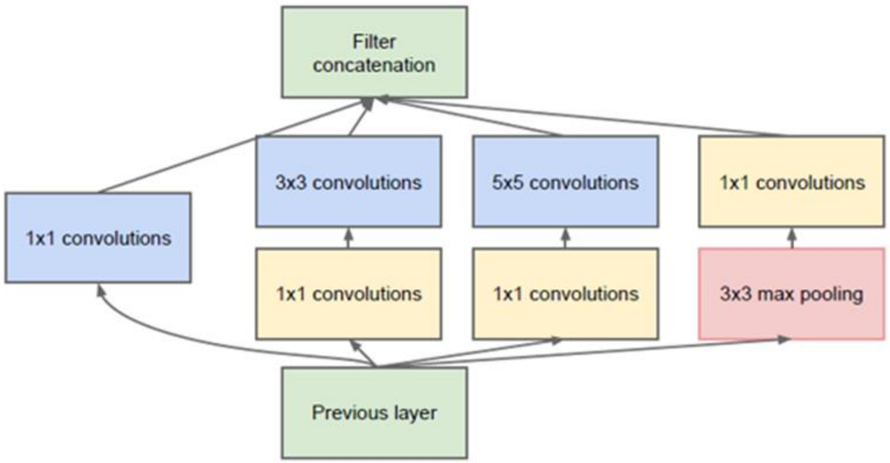
Inception model (a.k.a GoogLeNet)

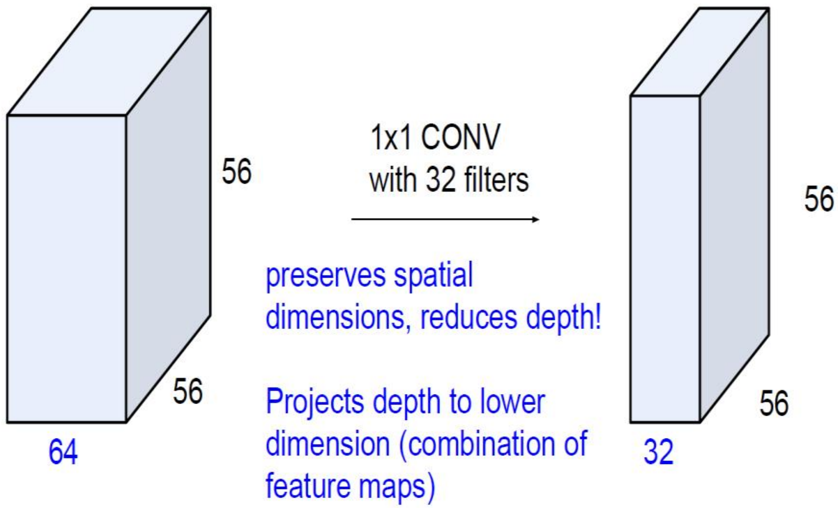


Source: Going Deeper with Convolutions (CVPR 2015)

- Deeper network with computational efficiency
  - ImageNet Large Scale Visual Recognition Competition (ILSVRC) 2014 winner (6.7% top 5 error)
  - 22 layers with 5 million parameters (12x less than AlexNet *ILSVRC 2012 winner)
  - Efficient “Inception” module

Inception module





Inception module

1x1 convolution

Source: Going Deeper with Convolutions (CVPR 2015)

- Local network topology composing the Inception model
  - Apply parallel filter operations on the input from previous layer
  - Multiple filter sizes for convolution (1x1, 3x3, 5x5)
  - 1x1 convolution for dimensionality reduction

Residual Neural Network (ResNet)

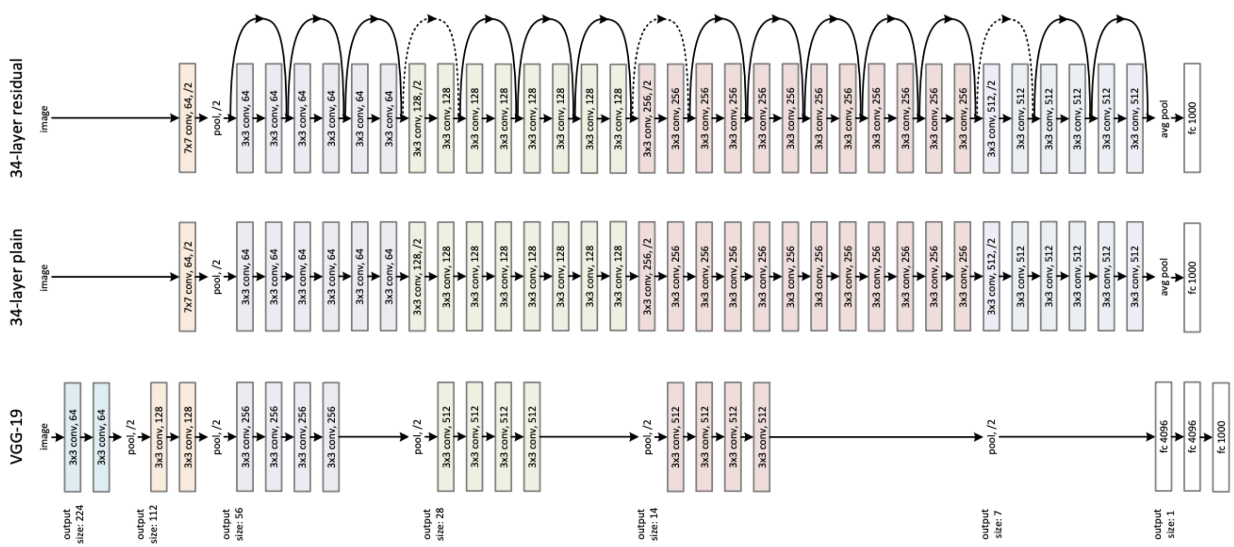
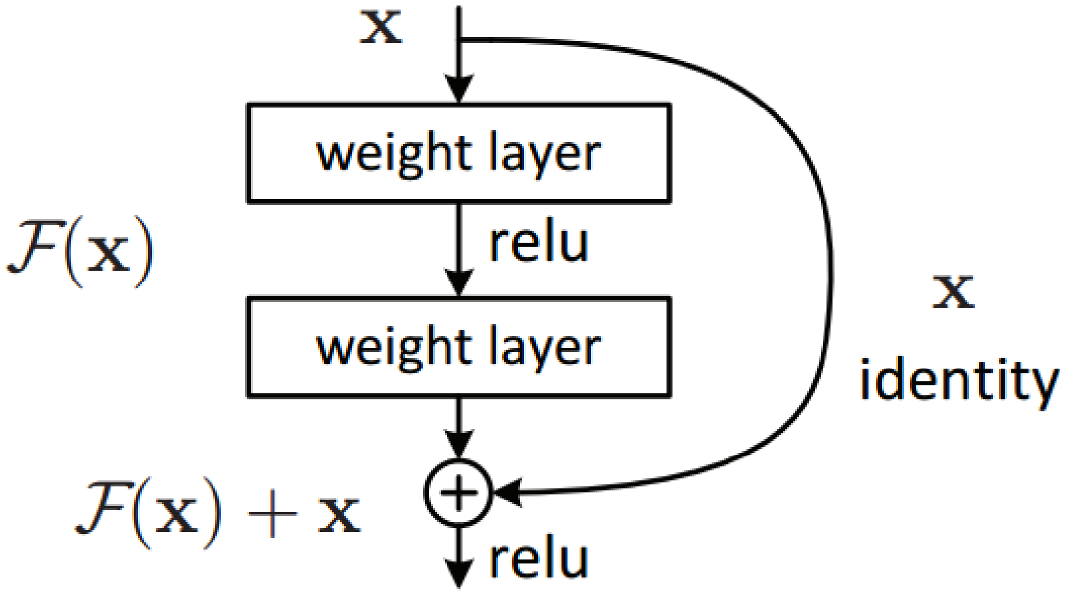


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

Source: Deep Residual Learning for Image Recognition (CVPR 2016)

- Deeper network with Residual connections
  - ImageNet Large Scale Visual Recognition Competition (ILSVRC) 2015 winner
  - In a ResNet, each layer's input is added to its output, creating a "shortcut connection" or a "skip connection."

Residual Block



Source: Deep Residual Learning for Image Recognition (CVPR 2016)

- Deeper network with Residual connections
  - Residual connection addresses the vanishing gradient problem by introducing shortcut connections
  - The original input is added to the output of the convolutional layers.

Assignment1_2 Objective (Transformer)

- Problem: Implementing Transformer from scratch
  - To understand Transformer architecture
  - Implement 4 components of the Transformer

    (1) Positional Encoding

    (2) Multi-head attention

    (3) Encoder

    (4) Decoder

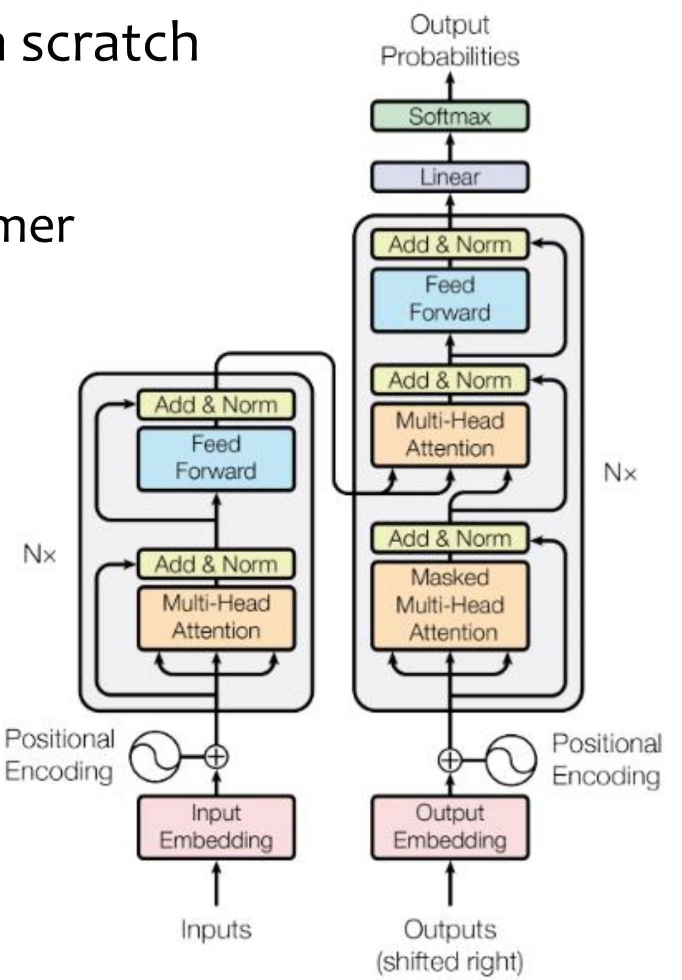


Figure 1: The Transformer - model architecture.

Source: Attention is all you need (NeurIPS 2017)

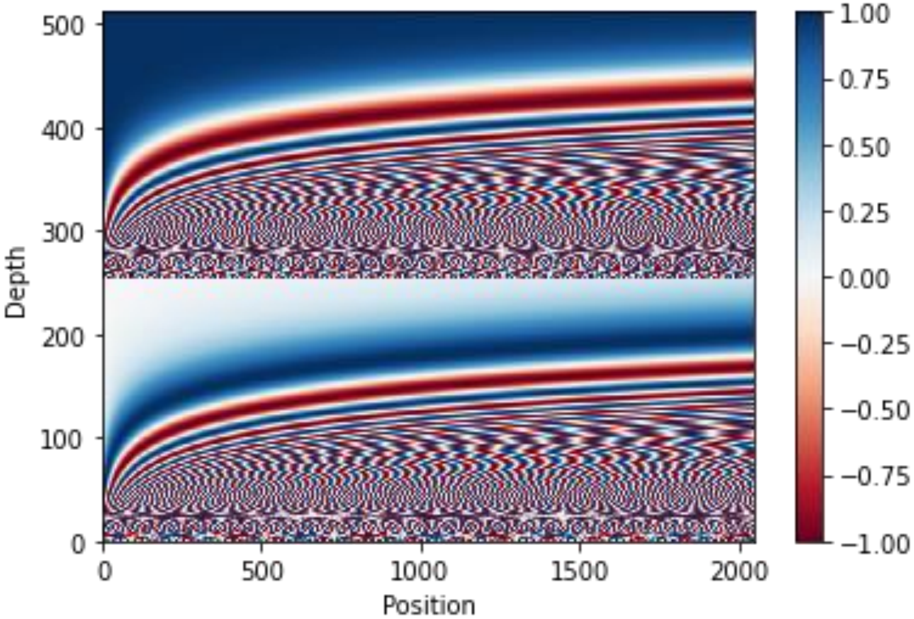
Implementing Transformer from scratch

- Four key components of the Transformer
  - Positional Encoding
  - Multi-head attention
  - Encoder
  - Decoder

- Positional Encoding
  - Means to convey order information

$$PE_{(pos,2i)} = sin(pos/10000^{2i/dim})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/dim})$$

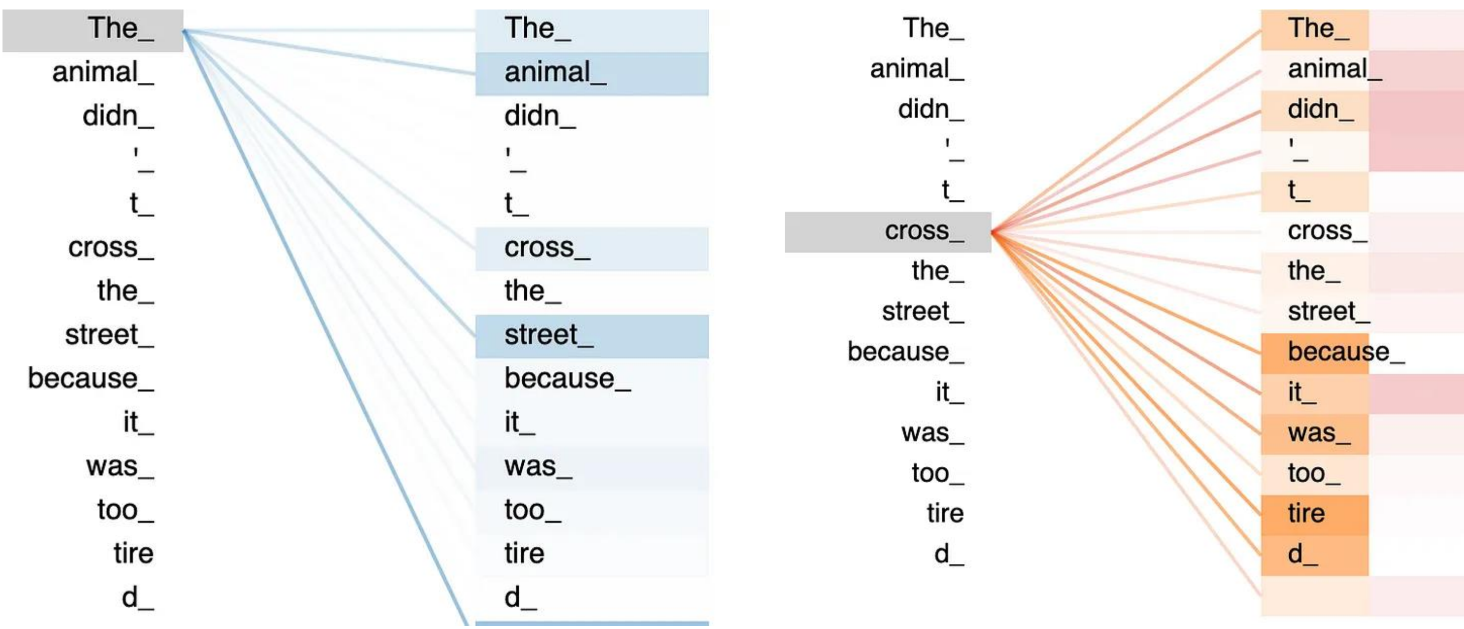
  - If we draw the above equation, it looks like the left image



Source: TensorFlow tutorial: Transformer model for language understanding

Implementing Transformer from scratch

- Multi-head attention layers learn various features of a sequence
  - Which do you like better, coffee or tea? Sentence type
  - Which do you like better, coffee or tea? Noun
  - Which do you like better, coffee or tea? Relationship
  - Which do you like better, coffee or tea? Emphasis



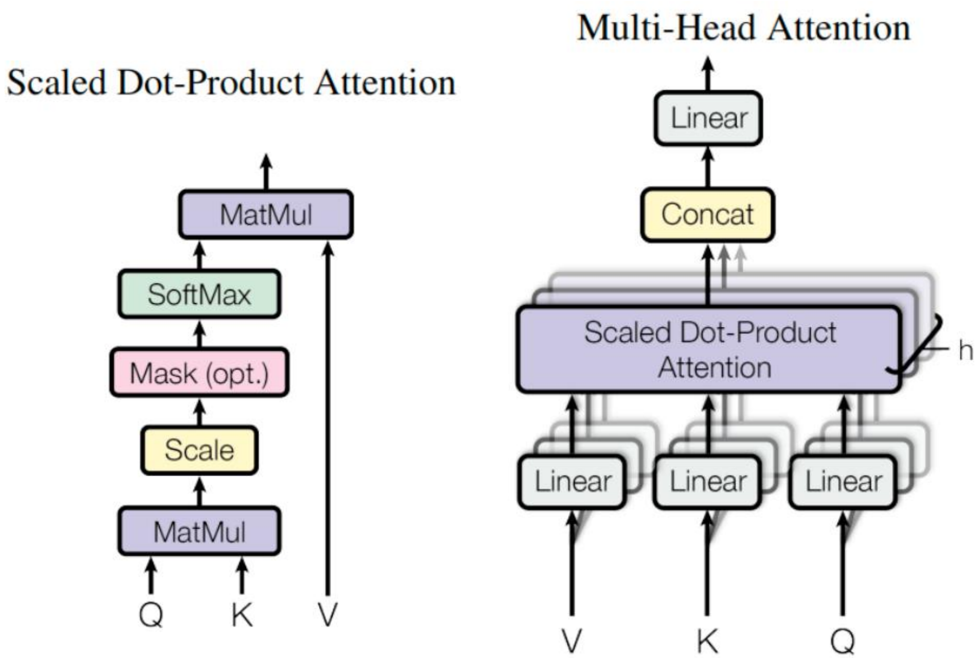
Source: Towards data science (Basics of self-attention)

Implementing Transformer from scratch

- Specific equation of multi-head attention layer

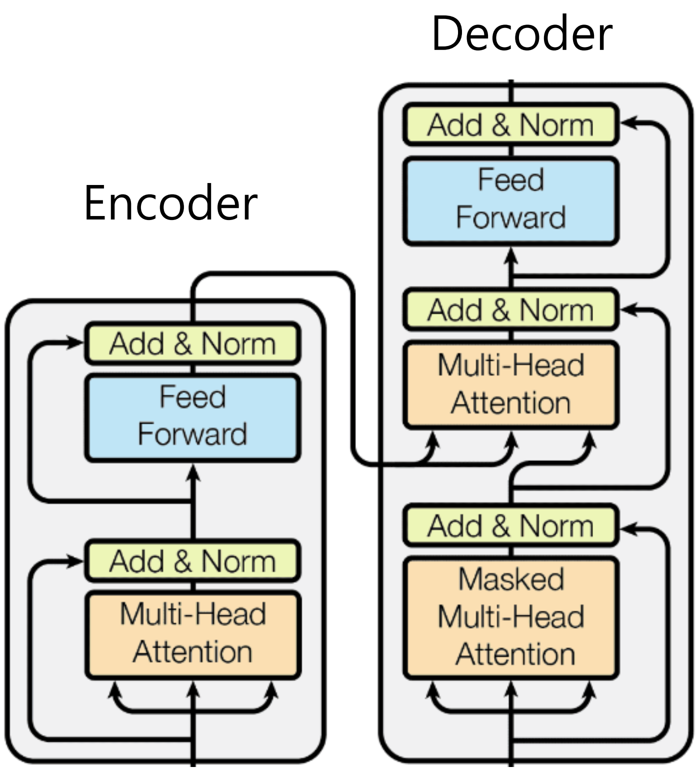
$$Q = X * W_q$$
$$K = X * W_k$$
$$V = X * W_v$$

$$scores = \frac{QK^T}{\sqrt{word\_dim}}$$
$$masked\_scores = mask(\frac{QK^T}{\sqrt{word\_dim}})$$
$$probs = softmax(masked\_scores)$$
$$heads = probsV$$
$$output = heads * W_o$$



Source: Attention is all you need (NeurIPS 2017)

Implementing Transformer from scratch



Source: Attention is all you need (NeurIPS 2017)

- Encoder
  - One Muti-head attention layer, one FFN layer, two normalization layers
- Decoder
  - Two Multi-head attention layers, one FFN layer, three normalization layers

Assignment_1-3 Objective (Vision Transformer)

- Problem 1: Part 2에서 구현한 Transformer를 참고하여 Vision Transformer(ViT) backbone의 code 구현
  - Image를 input으로 사용하는 특성을 고려하여 ViT 구조의 중요한 module 들을 각각 설명에 맞게 직접 구현

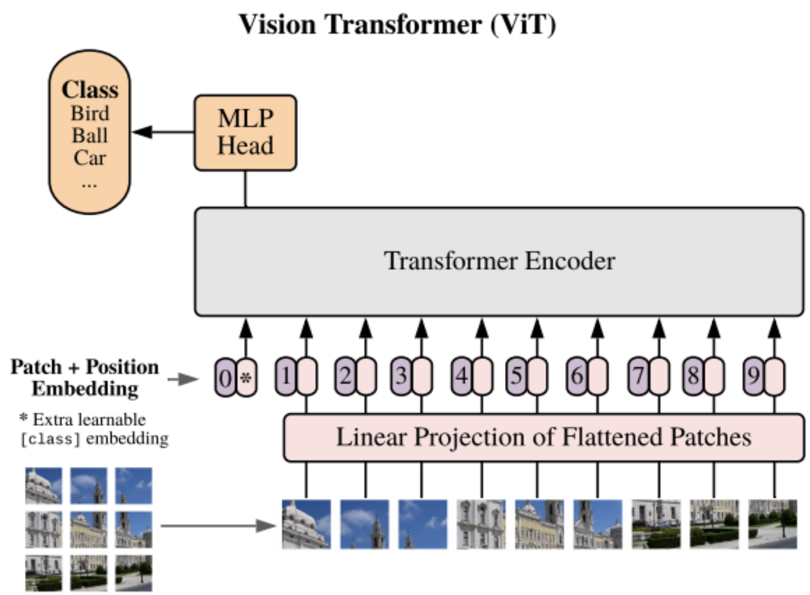
- Problem 2: 구현한 코드를 활용해 ViT model training (FashionMNIST) – Hyperparameter를 바꿔가며 FashionMNIST dataset에 대해 ViT model 학습 – 최소 85% 이상의 성능을 갖는 모델 학습
  - 최소 5번의 hyperparameter setting에 대해 실험 후, 각 실험에 대한 분석

FashionMNIST dataset

Source: https://www.kaggle.com/datasets/zalando-research/fashionmnist

- Each image in the FashionMNIST dataset is associated with a specific label, indicating the type of clothing item depicted in the image.
- Consists of 70,000 28x28 grayscale images in 10 classes, with 6,000 images per class
- There are 60,000 training images and 10,000 test images.

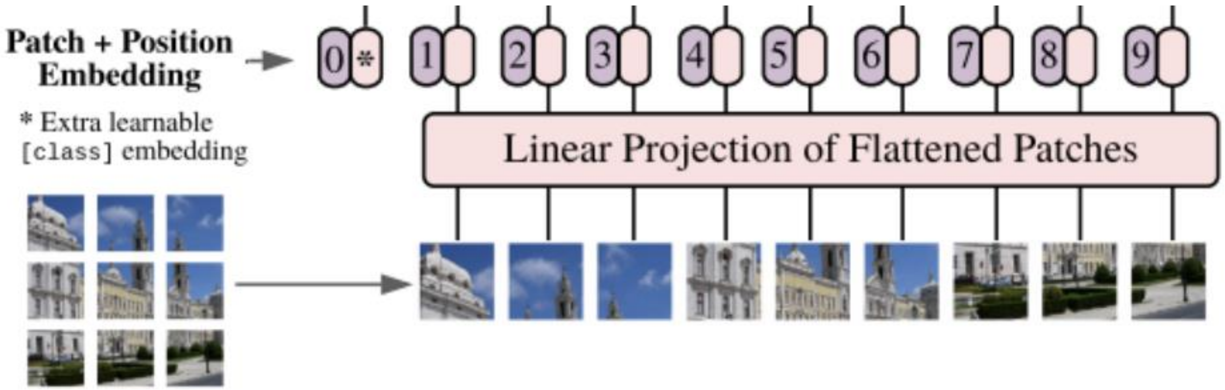
Vision Transformer (ViT)



Source: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale (ICLR 2021)

- ViT leverages the transformer architecture, which was originally proposed for natural language processing tasks.
  - Patchify image into patch embeddings for encoder input
  - Multi-head self-attention mechanism learns global relationships between image patches
  - MLP head performs target downstream task, e.g., classification.

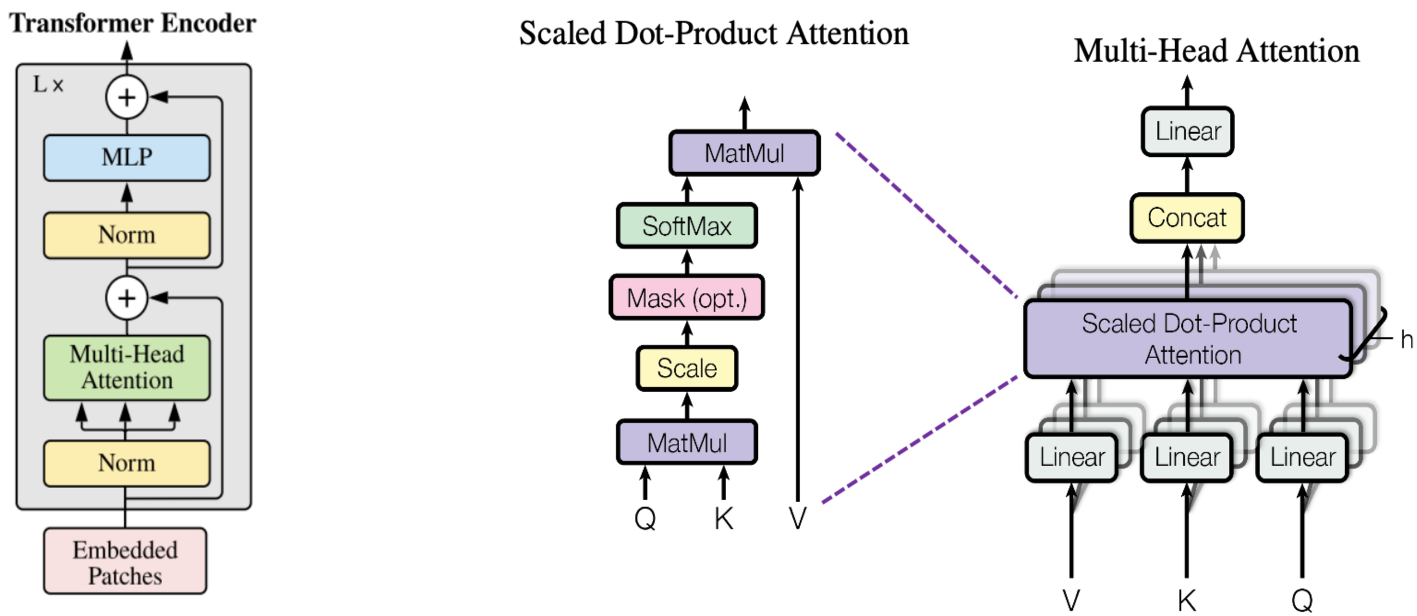
Patch Embed



Source: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale (ICLR 2021)

- **Patchify:**
  - Split the input image into small patches.
  - Flatten the patches and linearly project them into embedding dimension.
- **Class Token:** A special learnable token that is added to the sequence of patc h embeddings.
- **Learnable positional Encoding:** Instead of using fixed functions to compute positional encodings, ViT learns these encodings as part of the training proce ss.

Encoder (Multi-Head Attention & MLP)



Source: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale (ICLR 2021), Attention is All You Need (NeurIPS 2017)

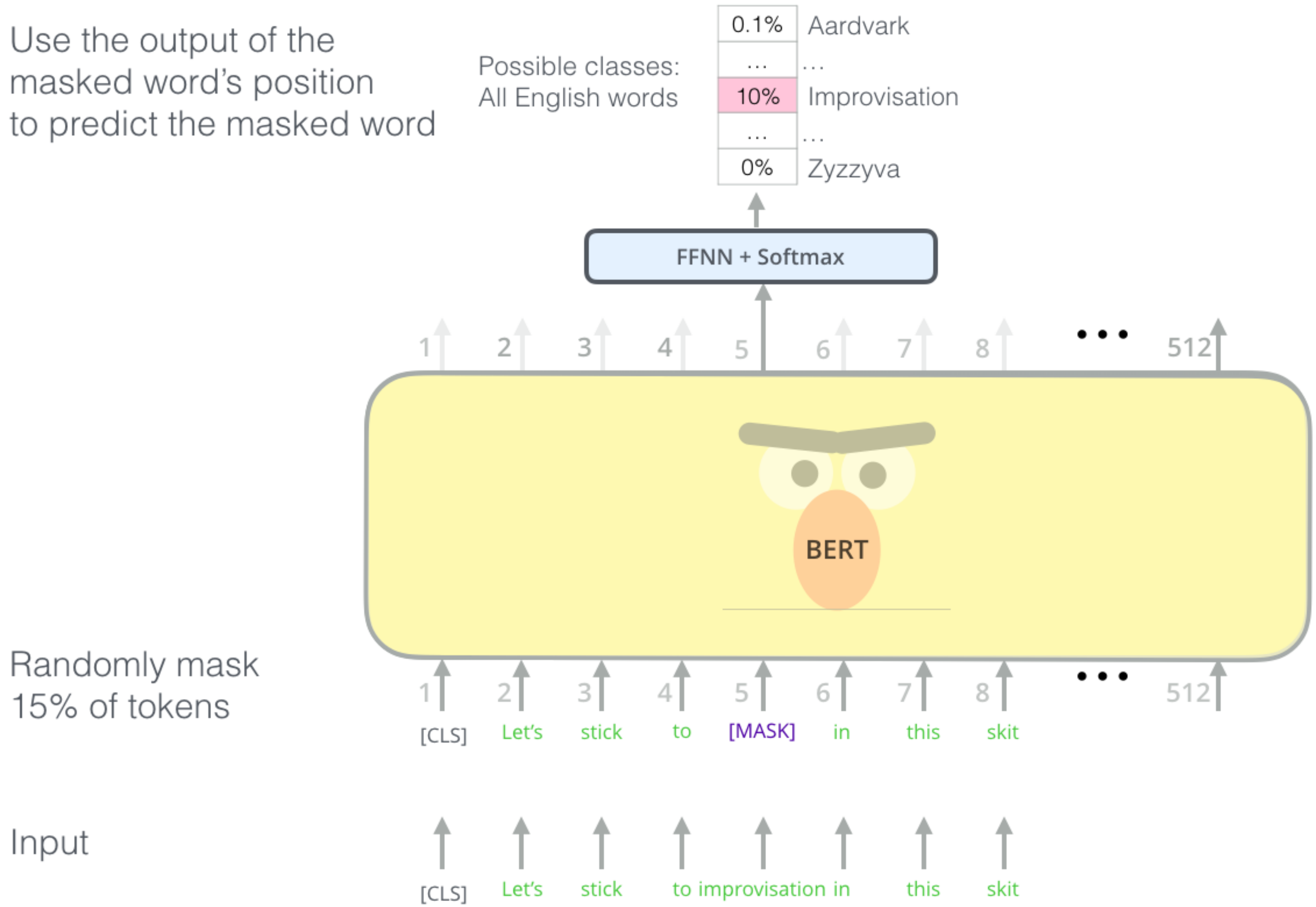
- **Query, Key, and Value Representati**ons: For each patch embedding, three vectors are derived: the Query (Q), Key (K), and Value (V) vectors.
- **Self-Attention Computation:** For each patch, the self-attention mechanism computes a set of attention scores, indicating the importance of other patches concerning the current patch.
- **Weighted Sum (Attention Output):** The attention scores obtained from the softmax operat ion are used to compute a weighted sum of the Value vectors (V) of all patches.

Assignment1_4 Objective (BERT finetune)

- Problem: Finetune BERT
  - Learn how to import a pre-trained model
  - Design the classification head for the downstream tasks

Finetune BERT

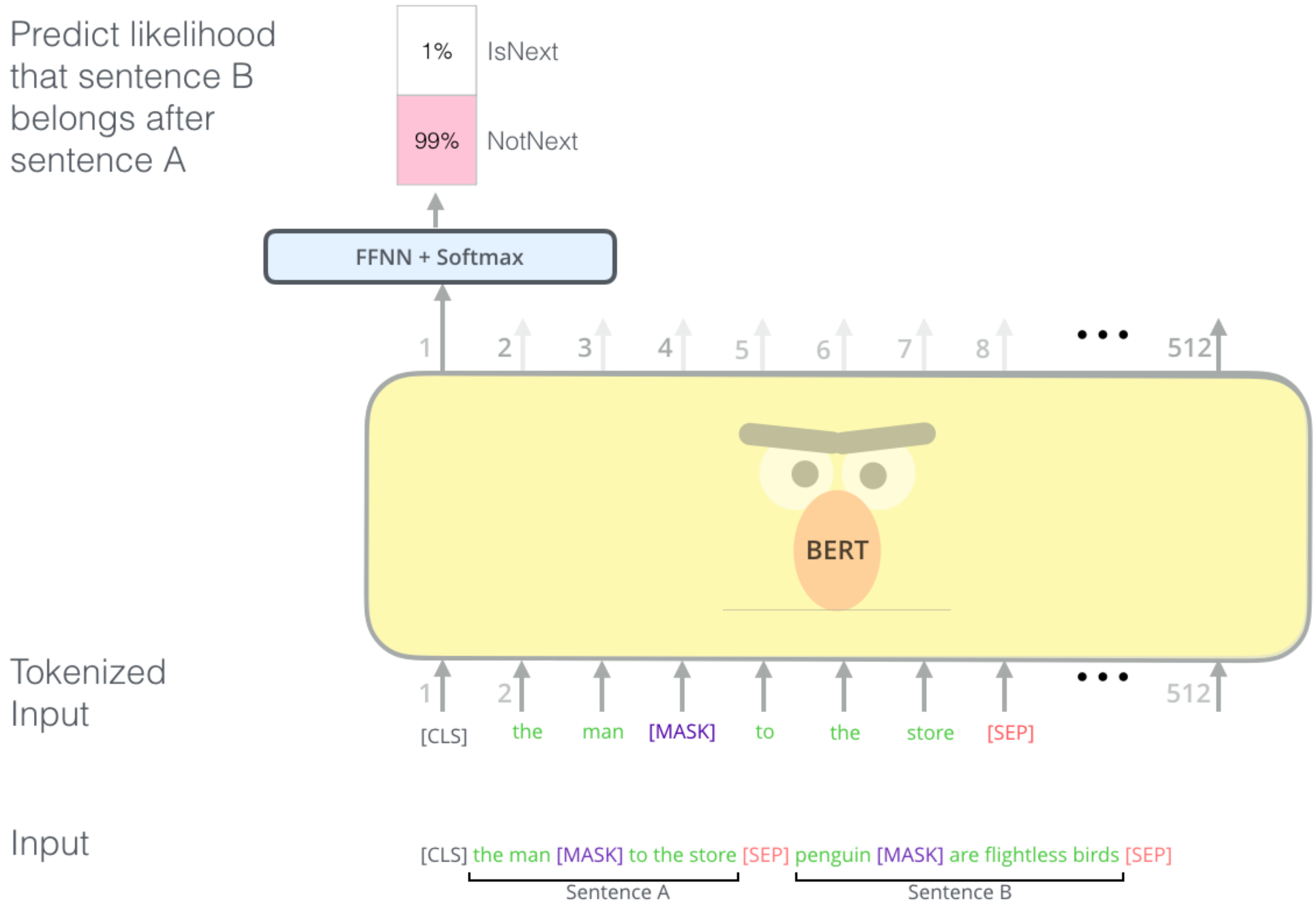
- Pre-trained language model: BERT
  - MLM (Masked Language Model)



Source: http://jalammar.github.io/illustrated-bert/

Finetune BERT

- Pre-trained language model: BERT
  - NSP (Next Sentence Prediction)



Source: http://jalammar.github.io/illustrated-bert/

Finetune BERT

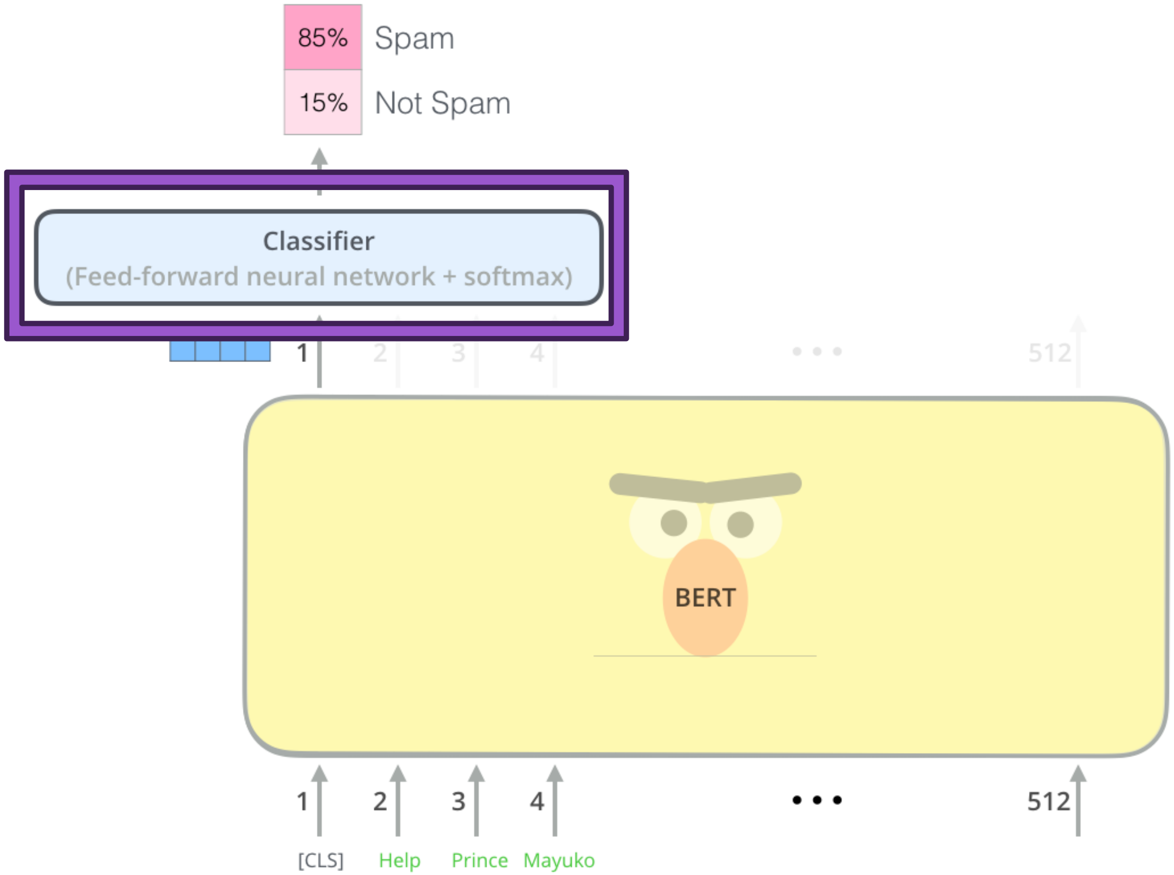
- IMDB dataset
  - Movie review classification tasks
  - Contains 50k movie reviews & sentiments
  - Predict the corresponding sentiments of the movie reviews

review	sentiment
49582 unique values	**2** unique values
One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. The...	positive
A wonderful little production. The filming technique is very unassuming- very old-time-B...	positive
I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air con...	positive
Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet & his par...	negative

Source: Learning Word Vectors for Sentiment Analysis (ACL 2011)

Finetune BERT

- Evaluation
  - Design classification head
  - Get text classification test accuracy



Source: http://jalammar.github.io/illustrated-bert/

Score criteria

- Assignment_1-1 : 25 points
  - Problem 1 : Simple CNN model training on CIFAR-10 dataset (5 points)
  - Problem 2 : Inception & Residual module 구현 (10 points)
  - Problem 3 : Inception module을 활용한 CNN model training (10 points)
- Assignment_1-2: 25 points
  - Problem: Implementing Transformer from scratch
  - Positional encoding, Multi-head attention, Encoder, Decoder, Final output (each 5 points)
- Assignment_1-3 : 25 points
  - Problem 1 : Vision Transformer(ViT) backbone의 code 구현 (15 points)
  - Problem 2 : 구현한 코드를 활용한 ViT model training (10 points)
- Assignment_1-4: 25 points
  - Problem 1: Design classification head (15pts)
  - Problem 2: Get text classification test accuracy bigger than 85% (10pts)

Output Examples

- Assignment_1-1
- Problem 1 : Simple CNN model training on CIFAR-10 dataset

```
print_accuracy(net, testloader)
```

```
Accuracy of the network on the 10000 test images: 55 %
```

- Problem 3 : Problem 2을 활용한 CNN model training (CIFAR-10)

```
[1,  2000] loss: 1.942
[1,  4000] loss: 1.616
[1,  6000] loss: 1.406
[1,  8000] loss: 1.278
[1, 10000] loss: 1.169
[1, 12000] loss: 1.088
[2,  2000] loss: 0.977
[2,  4000] loss: 0.936
[2,  6000] loss: 0.893
[2,  8000] loss: 0.830
[2, 10000] loss: 0.799
[2, 12000] loss: 0.762
Finished Training
Saved Trained Model
Accuracy of the network on the 10000 test images: 73 %
```

Output Examples

- Assignment_1-2
- Problem: Implementing Transformer from scratch

```
Epoch: 1, Loss: 8.676477432250977
Epoch: 2, Loss: 8.60608196258545
Epoch: 3, Loss: 8.547430038452148
Epoch: 4, Loss: 8.481799125671387
Epoch: 5, Loss: 8.42186450958252
Epoch: 6, Loss: 8.367515563964844
Epoch: 7, Loss: 8.309371948242188
Epoch: 8, Loss: 8.261741638183594
Epoch: 9, Loss: 8.216130256652832
Epoch: 10, Loss: 8.16081428527832
Epoch: 11, Loss: 8.11362075805664
Epoch: 12, Loss: 8.061040878295898
Epoch: 13, Loss: 8.005115509033203
Epoch: 14, Loss: 7.954127311706543
Epoch: 15, Loss: 7.899129390716553
Epoch: 16, Loss: 7.847209453582764
Epoch: 17, Loss: 7.791110992431641
```

Output Examples

- Assignment_1-3
- Problem 2 : Vision Transformer model training (FashionMNIST)

```
Using device:  cuda (Quadro RTX 8000)
Training:   0%|          | 0/5 [00:00<?, ?it/s]
Training:  20%|██        | 1/5 [00:11<00:47, 11.78s/it]
Epoch 1/5 loss: 0.60
Training:  40%|████      | 2/5 [00:23<00:35, 11.79s/it]
Epoch 2/5 loss: 0.42
Training:  60%|██████    | 3/5 [00:35<00:23, 11.80s/it]
Epoch 3/5 loss: 0.40
Training:  80%|████████  | 4/5 [00:48<00:12, 12.38s/it]
Epoch 4/5 loss: 0.38
Training: 100%|██████████| 5/5 [01:02<00:00, 12.59s/it]
Epoch 5/5 loss: 0.36
Testing: 100%|██████████| 79/79 [00:00<00:00, 79.78it/s]
Test loss: 0.37
Test accuracy: 86.50%
```

Output Examples

- Assignment_1-4
- Problem: Finetune BERT

```
Test Accuracy: 0.8957
              precision    recall  f1-score   support

           0       0.90      0.88      0.89      4979
           1       0.89      0.91      0.90      5021

    accuracy                           0.90     10000
   macro avg       0.90      0.90      0.90     10000
weighted avg       0.90      0.90      0.90     10000
```

How to install assignment files

- 포함된 파일: 7개
  1. Assignment1-1_CNN.ipynb
  2. Assignment1-2_ViT.ipynb
  3. Assignment1-3_Transformer_from_scratch.ipynb
  4. Assignment1-4_Finetuning_w_BERT.ipynb
  5. CollectSubmission.sh
  6. imgs, data 폴더 (data 폴더는 empty)

- 다운 후 설치 방법
  1. $tar zxvf Assignment1.tar.gz (decompress tar.gz file)
  2. $cd Assignment1
  3. $chmod 755 CollectSubmission.sh (get permission of script file)
  4. $conda activate {가상환경명}
  5. $jupyter notebook

- IPython notebook상에서 과제 수행

Important notes

- Due: 11/1 (수) 23:59
- PLEASE read the notes on the notebooks carefully
- Google first before mailing TAs

- Submitting your work
  - DO NOT clear the final outputs
  - After you are done all two parts:
    1. $ ./CollectSubmission.sh. 2000-00000 (학번)
    2. Upload the 2000-00000.tar.gz on eTL

.ipynb 4개
.pth 2개

- TA email : deeplearning.snu@gmail.com

FAQ_1

- Q : Batch size를 수정해도 되나요?
- A : 네. 모든 hyperparameter는 수정 가능 합니다.

- Q : n3xn3_blue나 n5xn5_blue는 어떤 인자인가요?
- A : n3xn3_blue나 n5xn5_blue는 conv_layer의 output channel 수 입니다. 본인 원하시는 대로 output channel 개수를 조정하셔서 구현하시면 됩니다.

- Q : 주어진 inception module내에서 concat을 해주는데, 이때 tensor 크기를 고려해야 하나요?
- A : 주어진 모듈 내 forward의 리턴값을 보시면, torch.cat으로 y1,y2,y3,y4를 dimension 1에 대해 concat 해줍니다. PyTorch에서는 tensor를 batch x channel x height x width 순으로 정의하기 때문에 코드에서는 y1~4를 channel에 대해 concat 해주는 것입니다. 따라서 channel개수를 제외한 batch, height, width의 크기만 맞춰주시면 됩니다.

FAQ_2

- Q : pth 파일이 무엇인가요?
- A : 학습 후 저장되는 모델 파일입니다. 학습을 완료하시면 자동생성됩니다.

- Q : max iteration 수나 data loader의 batch size, num_worker 수 등도 변경가능한지요?
- A : 가능합니다. 다만 batch size와 num_worker는 실험의 시간적인 면에서만 영향이 있고, 성능에는 영향이 거의 없을 것입니다.
- + epoch도 마찬가지로 기존2에서 더 늘려도 됩니다.

- Q : 코드 구현을 설명하는 부분도 점수에 포함되나요?
- A : ipynb 파일 마지막에 better_net에 관해 설명하는 부분은 copy 확인용이고 점수에 포함되지는 않습니다.
- 다만 Assignment 2-2의 3문제에 대한 답은 점수에 포함됩니다.

- Q : code 처음에 !는 무엇인가요?
- A : ! 뒤에 code는 shell script입니다. 즉, terminal에서 해당 code 돌리기 위해 사용되어집니다.


Thank You!