2024-2025 SPRING SEMESTER

Introduction to Robotics

-EE422/CS421-

VISION LANGUAGE MODELS FOR ROBOTIC PERCEPTION

WEEK 4

AGAH KUTAY FASTING

EMIN HAN CELIK

TIMUR GULMEZ

MELİH EGREK

Let me briefly and clearly explain the transition process from pre-training to fine-tuning with the CLIP model:

1. **Pre-Training Phase (Pre-preparation phase)**:
   CLIP (Contrastive Language-Image Pretraining) is trained by *contrastive learning using* a large-scale data set **(e.g. image-text pairs collected from the internet).** The goal is to establish a meaningful relationship between images and texts. The model *learns to match the textual description of an image* and **creates a generalized representation**. At this stage, training is usually done from scratch and the cost of calculation is high.

2. **Transition to Fine-Tuning: Fine-tuning** is done to adapt the pre-trained CLIP model **to a specific task or data set**. The migration takes place in these steps:

   - **Target Data Set Determination**: A labeled data set is prepared for a specific task (e.g. image classification, object detection).
   - **Starting Weights of the Model: The weights of the** pre-trained CLIP model are taken. These weights contain generalized information.
   - **Adaptation**: The model is retrained on the target data set with a low learning rate. This **allows task-specific features to be learned while retaining generalized knowledge.**
   - **Loss Function and Optimization**: A loss function (e.g., cross-entropy) is selected for the task, and the model is updated with an optimizer such as SGD or Adam.
   - **Dull Layers (optional):** *In some cases, certain layers of the model (e.g., early layers) are frozen and only the top layers are fine-tuned. This reduces the cost of calculation and avoids overfitting.*

3. **Different Approaches**:
   - **Full Fine-Tuning**: All model parameters are updated, but this requires more data and calculations.

- **Lightweight Fine-Tuning**: Only a few layers or an additional layer of adaptation (e.g. a linear head layer) are trained.
- **Prompt Tuning**: On the text side, task-specific prompts are optimized, model weights remain constant.

4. **Practical Notes**:
   - **Amount of Data**: Less data is sufficient for fine-tuning because the pre-trained model has already learned strong representations.
   - **Risk of Overfitting**: For small data sets, regularization (e.g., dropout, weight decay) or data augmentation is important.
   - **Evaluation**: After fine-tuning, the model is tested on the target task and performance metrics (accuracy, F1 score, etc.) are analyzed.

The main reason for using "text for speech" and "pixel for vision" in the CLIP model is to be able to represent different modes (text and image) in a common embedding space. **In this way, we can transfer, learn and match between text and image.**

Text Encoder

- **Definition**: Processes text data into high-dimensional vectors
- **Why use**: Encodes textual information in the form of numerical and meaningful vectors
- **Structure**: Usually **Transformer-based models (eg. BERT-like)**

Image Encoder

- **Definition**: Processes pixel data into high-dimensional vectors
- **Why use**: Encodes visual information in the form of numerical and meaningful vectors
- **Structure**: Usually **CNN or Vision Transformer architectures**

A Simple Example

Let's say there is an image of a "red apple" and the text "red apple":

1. **Image Encoder**: Takes the pixels of the apple image and converts it into a vector: [0.7, 0.2, 0.5, ...]
2. **Text Encoder**: Takes the text "red apple" and converts it into a vector: [0.65, 0.25, 0.48, ...]

CLIP training is optimized so that these two vectors are close to each other. Thus, the relevant text and images are located close to each other in the embedding space.

Through this common embedding space :

- **We can find the image that best suits a text**
- **We can find the text that best suits an image**
- *We can do transfer learning between different modes*

This structure is of great benefit in various applications such as image search, image classification, and text-image matching.

Self-attention

Self-attention is an important mechanism, especially in natural language processing and deep learning. This method allows each element in a sequence to update itself by considering its relationships with all other elements. Unlike traditional methods, self-attention evaluates the entire sequence together to learn the context of each element. For example, each word in a sentence obtains a more meaningful representation by considering the contributions of other words. This feature, especially as a key component of the Transformer model, increases parallel processing capabilities and makes it possible to learn dependencies in long sequences more efficiently.

Short Report: MFCC - Mel-Frequency Cepstral Coefficients

Objective:

MFCC (Mel-Frequency Cepstral Coefficients) is a method used to extract features from audio signals. It is widely used in speech recognition, speaker identification, and sound classification

systems.

1. Why MFCC?

Human ears do not hear all frequencies equally. MFCC helps computers to hear sound more like a

human. It creates a compact and useful representation of the audio.

2. How It Works (Main Steps):

1. Framing & Windowing:

The audio is divided into small frames (usually 25 ms) with some overlap. A Hamming window is

applied to each frame.

2. FFT (Fast Fourier Transform):

Converts the time-domain signal to the frequency domain.

3. Mel Scale Mapping:

Frequencies are converted to the Mel scale using a logarithmic function that mimics human

hearing.

4. Filterbank Processing:

Triangular filters are placed on Mel-scaled frequencies to extract energy from each band.

5. Log & DCT (Discrete Cosine Transform):

The log of filterbank energies is calculated, then DCT is applied to get the final MFCCs.

3. Output:

The result is a sequence of low-dimensional vectors (usually 13 coefficients per frame), which can be used as input for machine learning models.

4. Advantage of MFCC:

- No need for large datasets or pretraining

- Easy to implement

- Works well for real-time audio classification

Conclusion:

MFCC is a simple and effective method to represent audio signals for machine learning. It allows

machines to recognize speech and sound in a way that is inspired by human perception.

Residual Block Architecture

Residual blocks is building block of ResNet. It was introduced to solve the vanishing gradient problem in deep neural networks, which makes training very deep networks difficult.A residual block tries to learn a residual function instead of a direct mapping. That means:Instead of learning the output H(x) directly from input x.Also Residual Blocks used for detect important features.And this features can be carried forward unchanged.

Batch Normalization

Batch Normalization is a technique that:

- Normalizes the output of a layer (like a conv layer or a dense layer)
- Makes training faster, more stable, and less sensitive to initialization

With batch norm, they are rescaled to something like [0.1, -0.2, 0.3], and now gradients pass through safely — even in deep residual blocks.

Convolution Layer

This layer applies multiple filters (like 3x3 kernels) that slide over the image. These filters are trained to detect patterns such as edges, fur texture, or eyes of the cat.The result is a feature map — a transformed version of the image that highlights those patterns.

The result is a feature map — a transformed version of the image that highlights those patterns.

ReLU

ReLu is a activation function that adds non-linearity. ReLU only outputs 0 or positive numbers. If input is below 0 like -1 , output will be 0. If input greater than 0 than output will be the input.It’s very fast to compute.Solves vanishing gradient problem