**COURSE:** DSA4050

**COURSE TITLE:** Deep Learning for Computer Vision

**INSTRUCTOR:** DR. EDWARD OMBUI

# Week 10 Lesson Notes: Attention Mechanisms and Vision Transformers (ViT)

---

## Lesson Overview

Traditional deep learning models, such as Convolutional Neural Networks (CNNs), have been the backbone of computer vision for years. However, CNNs struggle with capturing long-range dependencies in images due to fixed-size convolutional filters.

This lesson focuses on attention mechanisms and Vision Transformers (ViTs), which have revolutionized deep learning by allowing models to focus on relevant features dynamically. Unlike CNNs, Transformers process entire images as sequences, making them more effective in capturing global context and long-range relationships in images.

### Key Topics Covered:

✓ **Attention Mechanisms** – What they are and why they matter.
✓ **Transformers in Deep Learning** – How they improve sequential processing.
✓ **Vision Transformers (ViT)** – How they apply transformers to computer vision.
✓ **Hands-on Implementation** – Training a **ViT model on an image dataset**.

By the end of this lesson, you will be able to:
✓ **Describe** the concept and importance of attention mechanisms in deep learning models.
✓ **Explain** the architecture and functionality of **Vision Transformers (ViT)**.
✓ **Implement and train** a Vision Transformer model in **TensorFlow or PyTorch**.

---

# 1 Attention Mechanisms in Deep Learning

## ◆ What is an Attention Mechanism?

Attention mechanisms allow deep learning models to **focus on the most important parts of input data** while processing information.

**Key Idea:** Instead of treating all input features equally, attention assigns **higher weights to relevant features** and **lower weights to less relevant features**.

## ◆ Why Are Attention Mechanisms Important?

✅ **Handles Long-Term Dependencies** – CNNs and RNNs struggle with long-range dependencies. Attention mechanisms help models focus on **important patterns** across large inputs.
✅ **Improves Efficiency** – Reduces the need for very deep networks by learning where to focus dynamically.
✅ **Enhances Interpretability** – Helps visualize which regions in an image or text the model is focusing on.

## ◆ Types of Attention Mechanisms

| Type | Description | Example Use Case |
|---|---|---|
| **Soft Attention** | Assigns different attention weights to all input regions | Image captioning |
| **Hard Attention** | Focuses on a single region at a time (non-differentiable) | Object detection |
| **Self-Attention** | Computes relationships between all input elements | Transformers (e.g., ViT, BERT) |
| **Multi-Head Attention** | Applies multiple attention layers in parallel for better feature learning | Vision Transformers |

📖 **Reference:** Bahdanau et al. (2014). *Neural Machine Translation by Jointly Learning to Align and Translate.*

---

# 2️⃣ Transformers: Revolutionizing Deep Learning

Transformers were originally introduced in **Natural Language Processing (NLP)** to replace Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs). They have since been adapted for computer vision through **Vision Transformers (ViTs)**.

## ◆ Transformer Architecture Overview

### Key Components of a Transformer:

✓ **Self-Attention Mechanism:** Determines how different parts of an input relate to each other.
✓ **Multi-Head Attention:** Uses multiple attention layers to capture diverse features.
✓ **Feedforward Network (FFN):** Processes information after attention layers.
✓ **Positional Encoding:** Adds sequence information since transformers do not have built-in order awareness.

### Mathematical Representation of Self-Attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Where:

- QQQ = Query matrix (features to search)
- KKK = Key matrix (reference features)
- VVV = Value matrix (actual data representation)
- dkd_kdk = Dimensionality of queries and keys

📖 **Reference:** Vaswani et al. (2017). *Attention Is All You Need.*

# 3 Vision Transformers (ViT)

## ◆ What is a Vision Transformer (ViT)?

**ViTs apply transformers to image processing by treating an image as a sequence of patches** instead of using convolutional layers.

## ◆ ViT Architecture

### Steps in ViT Processing:

1 Image Tokenization: The image is divided into small patches (e.g., 16×16 pixels).
2 Embedding: Each patch is flattened and projected into a feature vector.
3 Positional Encoding: Since Transformers do not have spatial awareness, positional

embeddings are added.
4️⃣ Transformer Encoder: The sequence of patch embeddings is processed through multi-head self-attention layers.
5️⃣ Classification Head: The final output is passed through a fully connected layer for classification.

| Feature | CNNs | ViTs |
|---|---|---|
| **Feature Extraction** | Uses local convolution filters | Captures long-range dependencies |
| **Global Context Awareness** | Limited | Stronger global feature representation |
| **Scalability** | Hard to scale beyond certain depths | Scales well with large datasets |

📖 **Reference:** Dosovitskiy et al. (2021). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.*

## 4️⃣ Implementing a Vision Transformer in TensorFlow/PyTorch

### ◆ Install Required Libraries

```
!pip install tensorflow transformers
```

### ◆ Implementing a Simple Vision Transformer in TensorFlow

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dense, LayerNormalization, MultiHeadAttention, Flatten
import numpy as np
# Define Transformer Block
class TransformerBlock(tf.keras.layers.Layer):
    def __init__(self, embed_dim, num_heads, ff_dim):
        super(TransformerBlock, self).__init__()
        self.att = MultiHeadAttention(num_heads=num_heads, key_dim=embed_dim)
        self.ffn = keras.Sequential([
```

```
            Dense(ff_dim, activation="relu"),
            Dense(embed_dim),
        ])
        self.layernorm1 = LayerNormalization(epsilon=1e-6)
        self.layernorm2 = LayerNormalization(epsilon=1e-6)

    def call(self, inputs):
        attn_output = self.att(inputs, inputs)
        out1 = self.layernorm1(inputs + attn_output)
        ffn_output = self.ffn(out1)
        return self.layernorm2(out1 + ffn_output)

# Create a sample image batch (Batch Size: 1, 32x32 Image with 3 Channels)
sample_image = np.random.rand(1, 32, 32, 3)
```

### ◆ Training ViT on an Image Dataset (CIFAR-10 Example)

```
# Load CIFAR-10 Dataset
(x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0  # Normalize
# Define ViT Model
model = keras.Sequential([
    TransformerBlock(embed_dim=64, num_heads=4, ff_dim=128),
    Flatten(),
    Dense(10, activation="softmax")  # 10-class classification
])
# Compile Model
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy",
metrics=["accuracy"])
# Train Model
model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))
```

## 5 Conclusion & Next Steps

✅ Attention mechanisms improve deep learning by allowing models to focus on important features.

✅ Transformers revolutionized sequential processing in AI, outperforming CNNs in long-range dependency tasks.

✅ Vision Transformers (ViTs) apply transformers to image recognition using patch embeddings.

✅ Practical Implementation: We trained a ViT model on an image dataset in TensorFlow.

## Next Steps:

🚀 Experiment with different ViT architectures (e.g., Swin Transformer, DeiT).

🚀 Try ViTs on custom datasets (e.g., medical imaging, satellite imagery).