Vision Transformer - 실습

이찬호
국방인공지능융합연구소

CONTENTS

1. DATASETS

❑ CIFAR-10 (Canadian Institute for Advanced Research-10)

- 해상도 : 32 X 32 X 3
- 클래스 : 10
- 학습 데이터 수 : 50,000장 (클래스 당 5,000장)
- 테스트 데이터 수 : 10,000장 (클래스 당 1,000장)

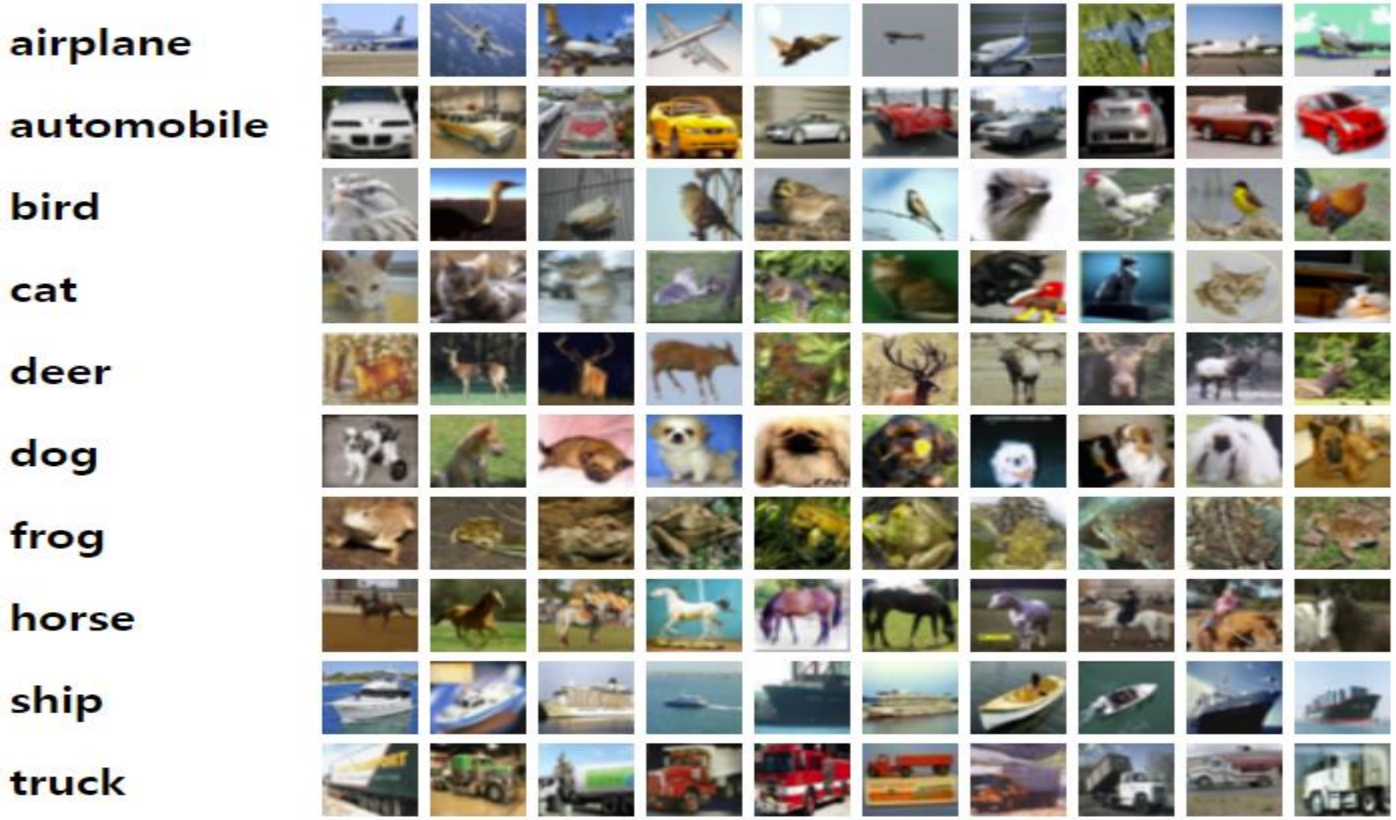


FIG 1. CIFAR-10 Datasets 일부

1. DATASETS

❑ ImageNet-100

- 해상도 : 다양함
- 클래스 : 100
- 학습 데이터 수 : 130,000장 (클래스 당 1,300장)
- 테스트 데이터 수 : 5,000장 (클래스 당 50장)

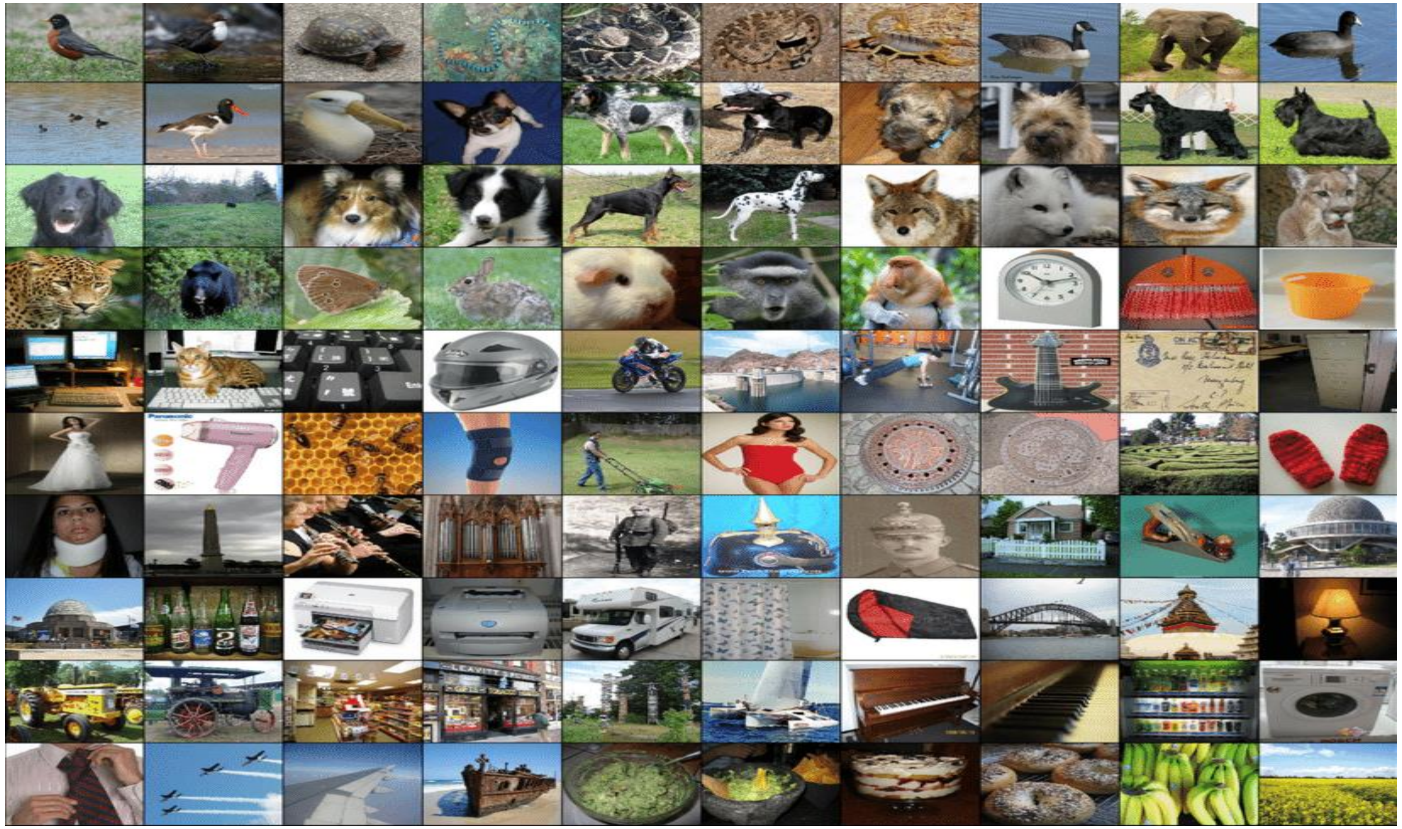


FIG 2. ImageNet-100 Datasets 일부

2. ViT ARCHITECTURE & CODE

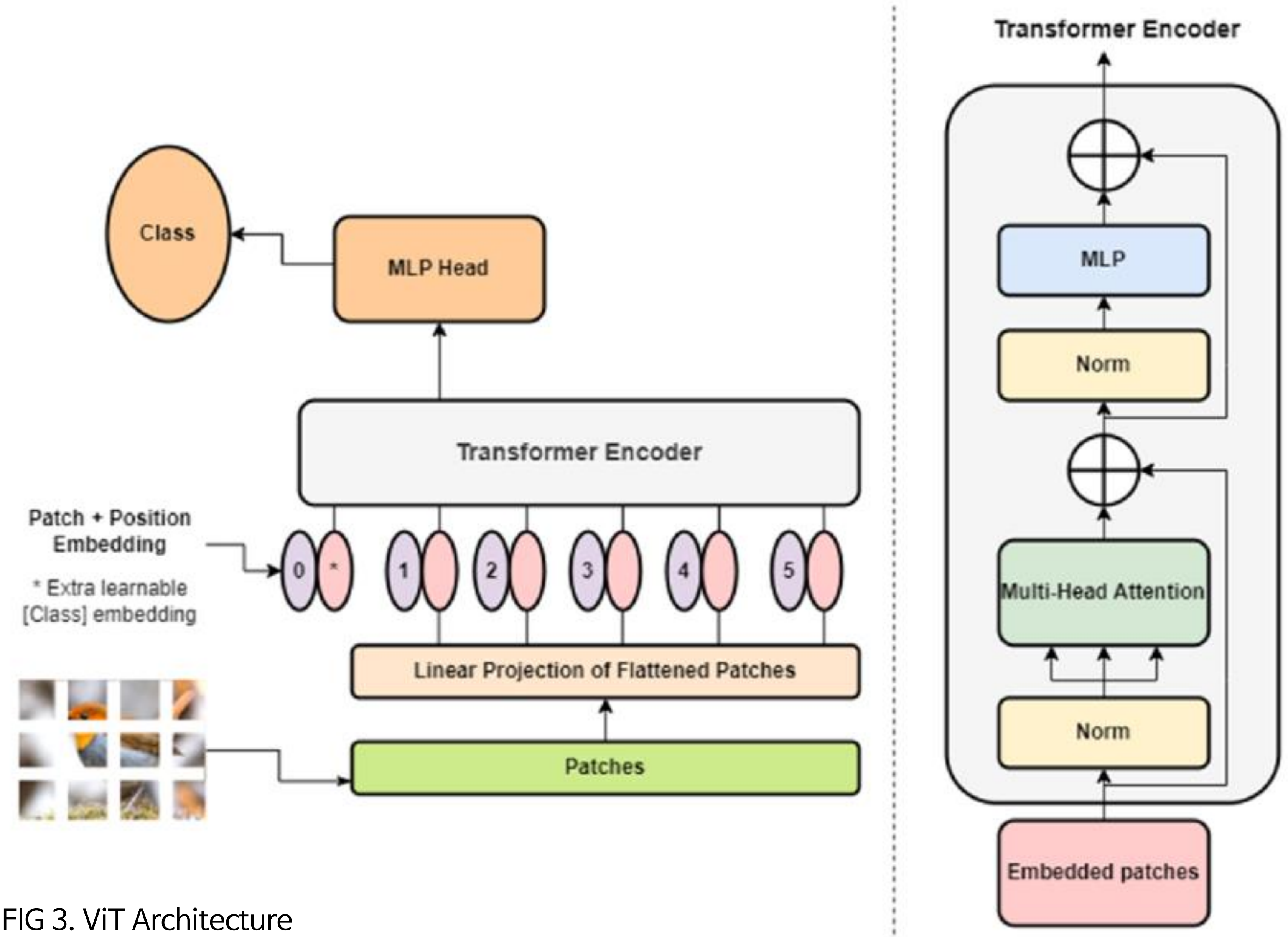


FIG 3. ViT Architecture

2. ViT ARCHITECTURE & CODE - create_patches

❑ create_patches

- 이미지를 작은 패치(patch)로 분할하여 2차원 형태로 반환

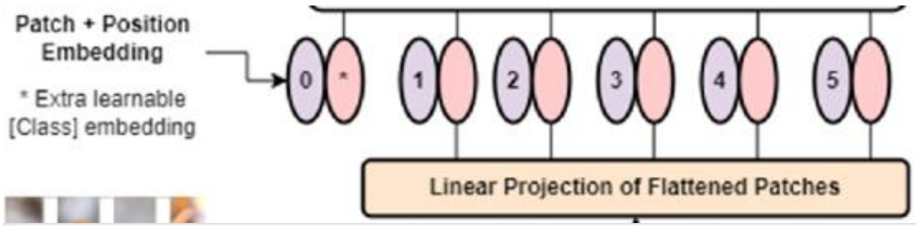

```
def create_patches(images, patch_size):
    patches = tf.image.extract_patches(
        images= images,
        sizes= [1, patch_size, patch_size, 1],
        strides= [1, patch_size, patch_size, 1],
        rates= [1, 1, 1, 1],
        padding= 'VALID')
    patches = tf.reshape(patches, [tf.shape(images)[0], -1, patch_size * patch_size * 3])
    return patches
```

FIG 4. patches code

2. ViT ARCHITECTURE & CODE - patchEncoder

❑ patchEncoder

- 패치를 입력 받아 인코딩(선형 변환)하여 고차원 벡터로 변환하고, 위치 임베딩을 추가




```python
class patchEncoder(layers.Layer):
    def __init__(self, num_patches, projection_dim):
        super(patchEncoder, self).__init__()
        self.num_patches = num_patches
        self.projection = layers.Dense(units= projection_dim)
        self.position_embedding = layers.Embedding(input_dim= num_patches, output_dim= projection_dim)

    def call(self, patch):
        positions= tf.range(start= 0, limit= self.num_patches, delta= 1)
        positions= tf.expand_dims(positions, axis= 0)
        encoded= self.projection(patch) + self.position_embedding(positions)
        return encoded

    def get_config(self):
        config= super(patchEncoder, self).get_config()
        config.update({'num_patches': self.num_patches, 'projection_dim': self.projection.units})
        return config
```

FIG 5. patcheEncoder code

2. ViT ARCHITECTURE & CODE – visionTransformer

❑ visionTransformer

- ViT 모델 정의

```
def visionTransformer(input_shape,
                      patch_size,
                      num_patches,
                      projection_dim,
                      num_heads,
                      num_transformer_layers,
                      mlp_head_units,
                      dropout_rate= .1,
                      num_classes= 10):
    inputs= layers.Input(shape= input_shape)
    patches= create_patches(inputs, patch_size)
    encoded_patches= patchEncoder(num_patches, projection_dim)(patches)

    for _ in range(num_transformer_layers):
        # Layer Normalization 1
        x1 = layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
        # Multi-Head Attention Layer
        attention_output = layers.MultiHeadAttention(num_heads=num_heads, key_dim=projection_dim)(x1, x1)
        # Skip Connection 1
        x2 = layers.Add()([attention_output, encoded_patches])
        # Layer Normalization 2
        x3 = layers.LayerNormalization(epsilon=1e-6)(x2)
        # MLP
        x3 = layers.Dense(units=projection_dim, activation=tf.nn.gelu)(x3)
        x3 = layers.Dropout(dropout_rate)(x3)   # Dropout 추가
        x3 = layers.Dense(units=projection_dim, activation=tf.nn.gelu)(x3)
        x3 = layers.Dropout(dropout_rate)(x3)   # Dropout 추가
        # Skip Connection 2
        encoded_patches = layers.Add()([x3, x2])

    # Layer Normalization & Global Average Pooling
    representation = layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
    representation = layers.GlobalAveragePooling1D()(representation)

    # MLP Head
    features = layers.Dense(units=mlp_head_units[0], activation=tf.nn.gelu)(representation)
    for units in mlp_head_units[1:]:
        features = layers.Dense(units=units, activation=tf.nn.gelu)(features)
    logits = layers.Dense(100)(features)   # ImageNet 100 클래스

    model = keras.Model(inputs=inputs, outputs=logits)
    return model
```

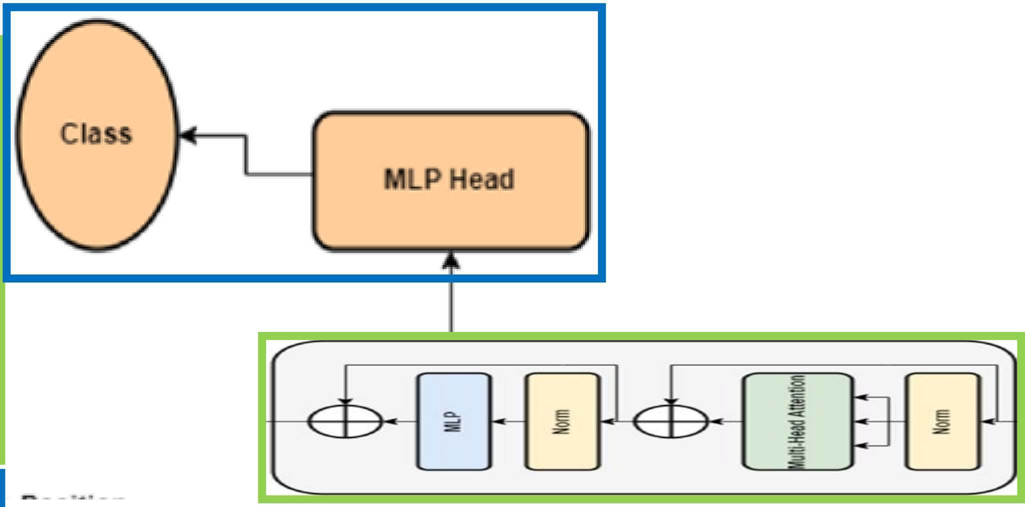
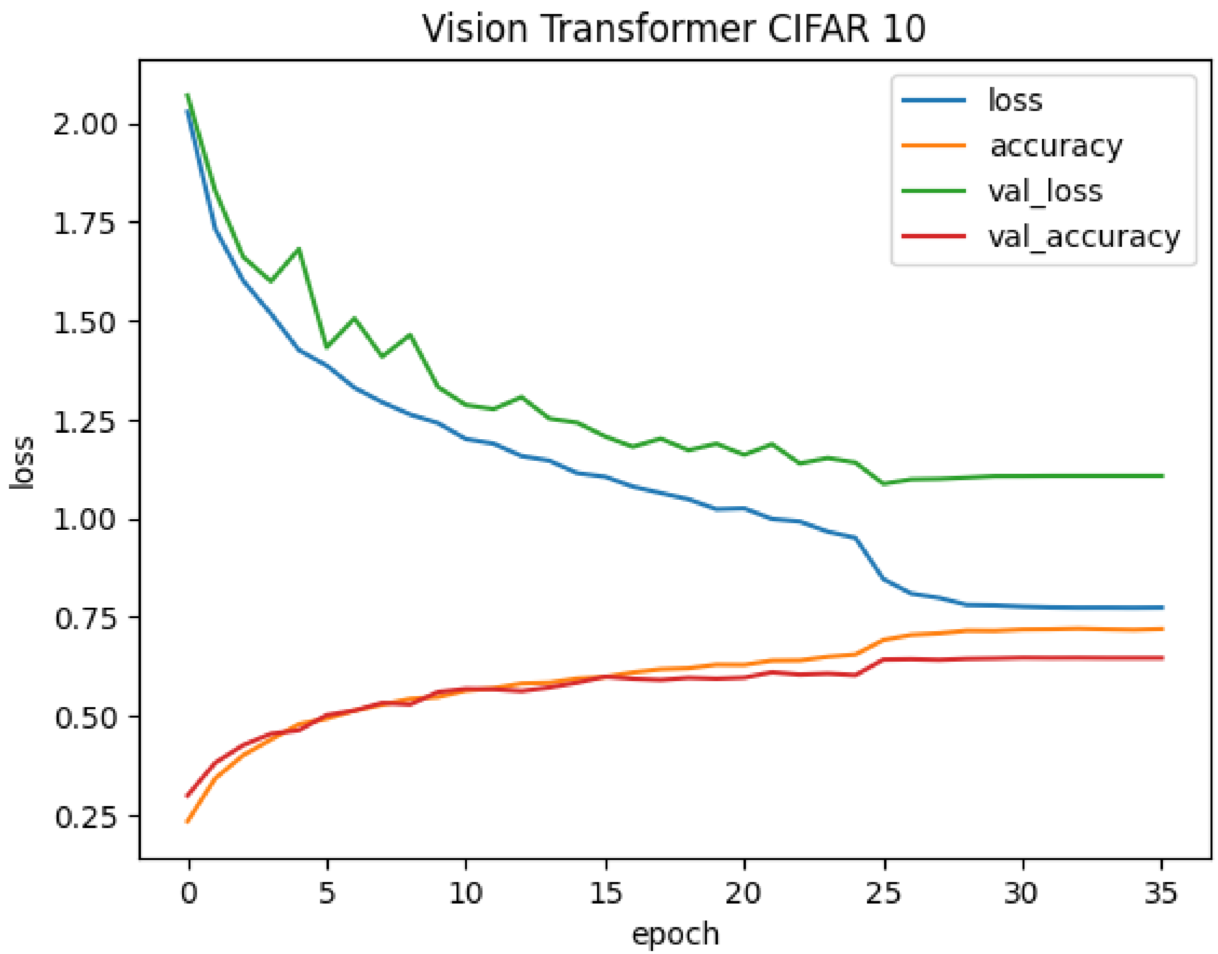


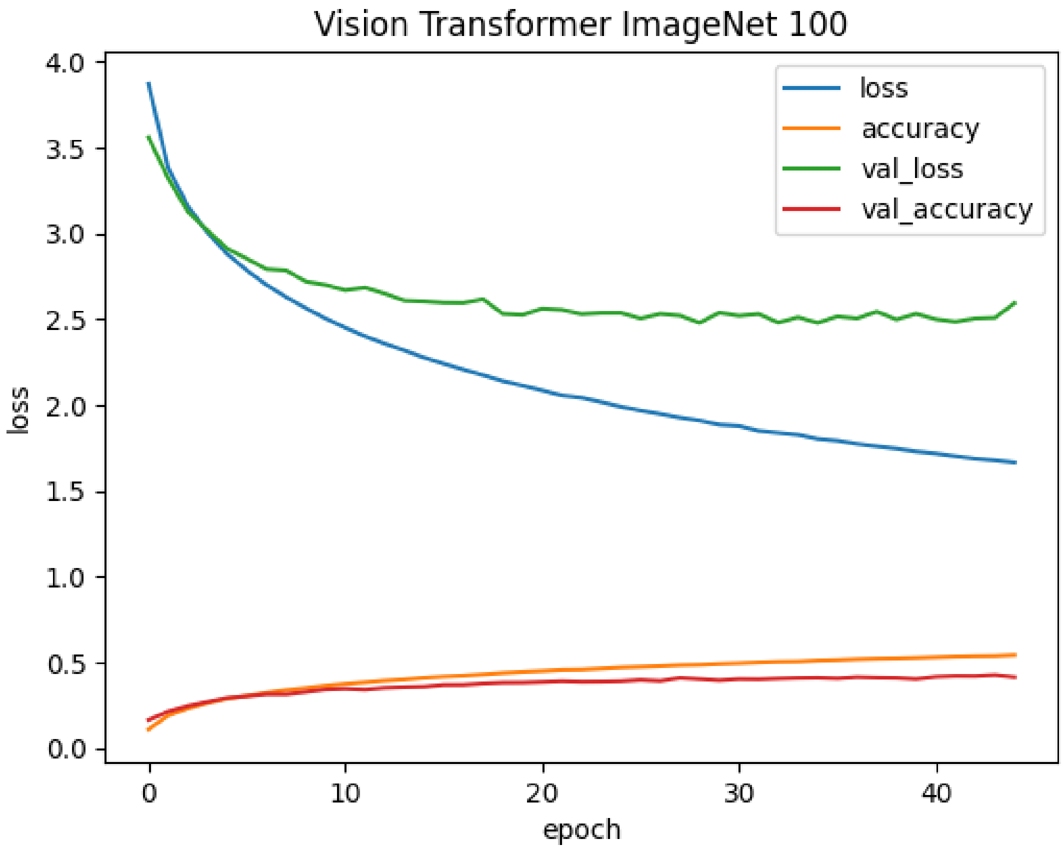
FIG 6. visionTransformer code

3. Result



Datasets	CIFAR-10
Epoch	36
Loss	0.8460
Accuracy	0.6919
Validation Loss	1.0870
Validation Accuracy	0.6422
Test Loss	1.1141
Test Accuracy	0.6273

3. Result



Datasets	ImageNet-100
Epoch	45
Loss	1.8012
Accuracy	0.5104
Validation Loss	2.4765
Validation Accuracy	0.4110

THANKS FOR YOUR ATTENTION

이찬호
국방인공지능융합연구소