Développez une preuve de concept

Modèle Vision Transformer (ViT)

OpenclassRoom

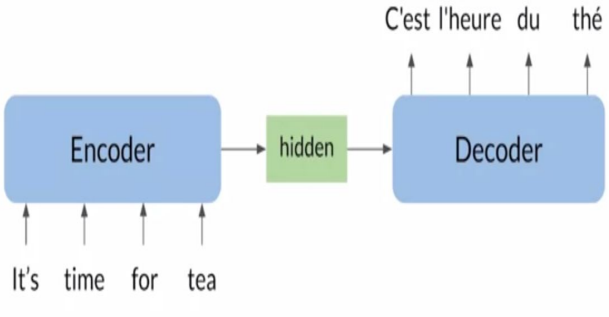
Benjamin Bouchard

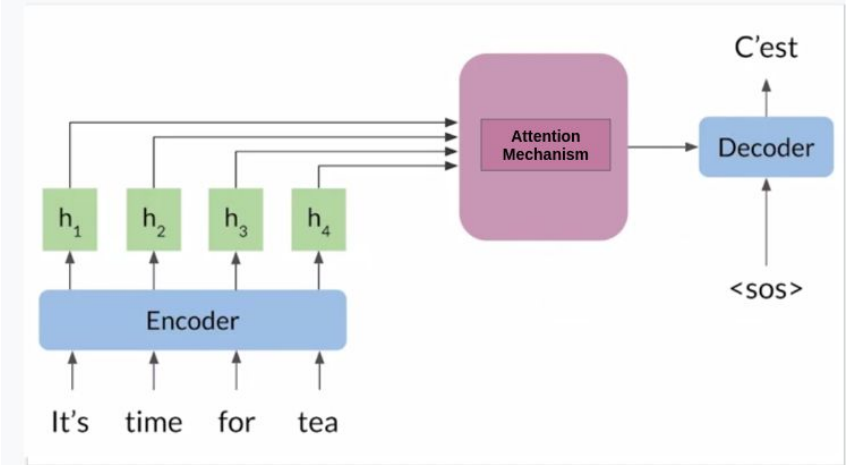
De la NLP à la Computer Vision

- Modèles **Transformer** => Révolution NLP
- Basés sur le mécanisme d'**Attention**
- Modèle Vision Transformer (**ViT**): Transformer de la NLP => Computer Vision

Mécanisme d'Attention

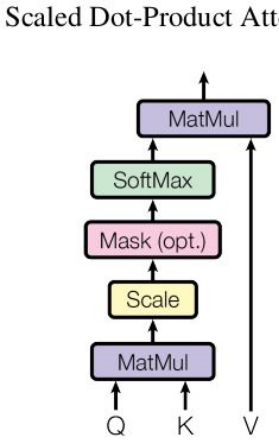
- Mécanisme d'Attention introduit en 2014 (Neural Machine Translation by Jointly Learning to Align and Translate) pour améliorer les modèles existants de transduction de séquences,





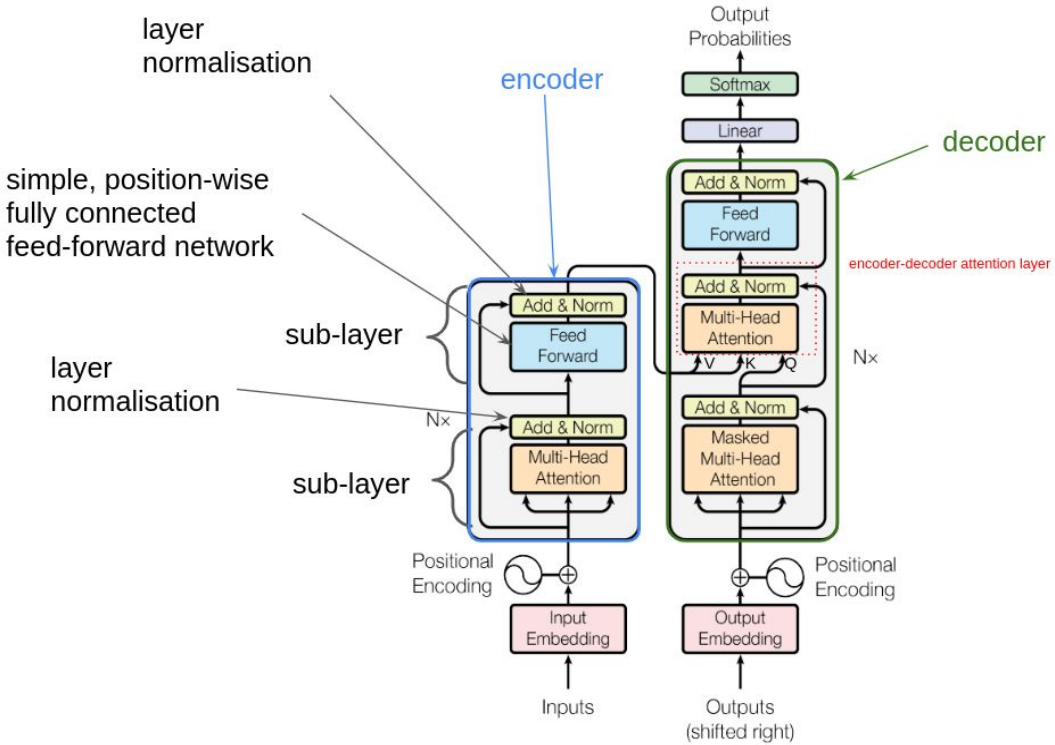
Attention et Self-Attention

- En 2017, Attention Is All You Need => self-Attention

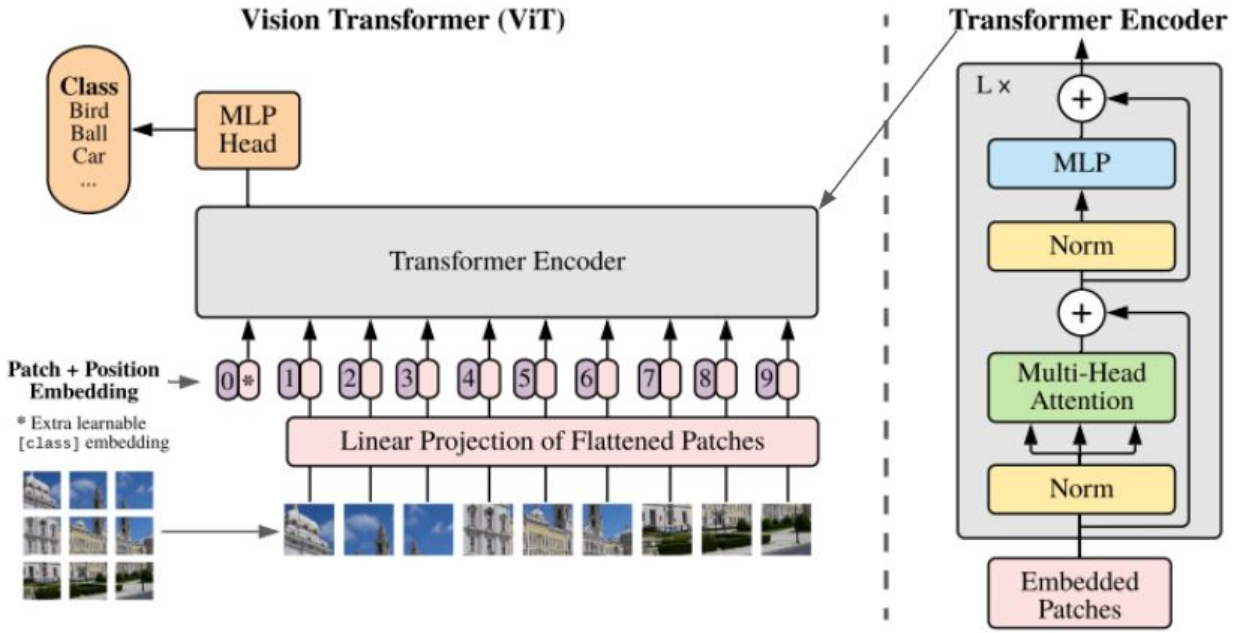


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Modèle Transformer



Modèle Vision Transformer



ViT: variants et benchmark

- Variants de ViT

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 1: Details of Vision Transformer model variants.

- Benchmark

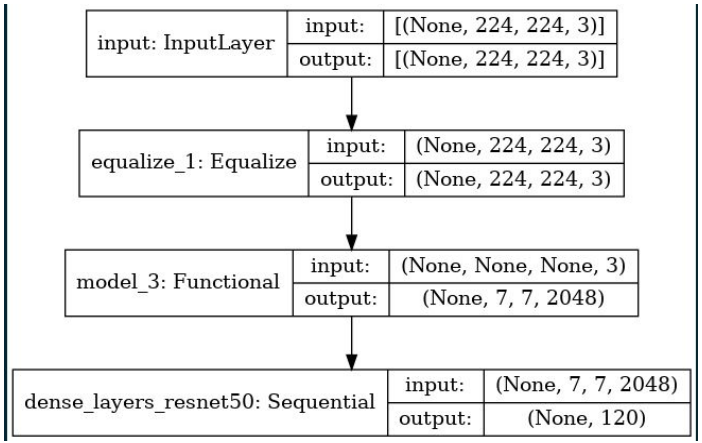
	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k

Benchmark ViT Vs ResNet50

- Data set Stanford Dogs du projet 6 issu de Image Net (> 14 m images)
- Mise en place d'un benchmark du modèle ViT face au modèle ResNet50
- Utilisation de la partie "feature extraction" des deux modèles pré-entraînés sur Image Net
- "Fine-Tuning" de la partie dense (en sortie)
- Hyper-tuning des hyper-paramètres dans les deux cas.

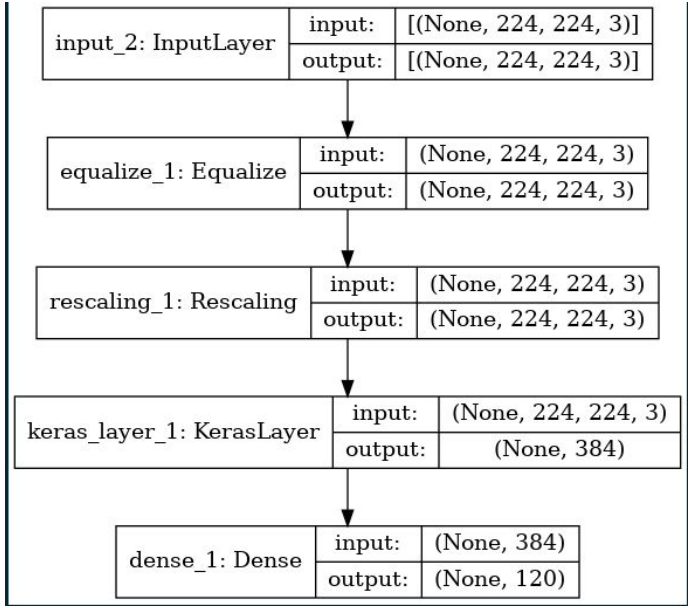
ResNet 50

```python
def my_pretrained_resnet50(hp,mode,num_classes):

    #pretrained_resnet50
    pretrained_resnet50 = get_pretrained_model(tfkapp.resnet50.ResNet50,tfkapp.resnet50.preprocess_input)

    # preprocess_layers
    preprocess_layer = tf.keras.models.Sequential(
        name= "preprocess_layer_resnet50",
        layers = [
            tf.keras.Input(name="input",shape=IMG_SIZE + (3,)),
            Equalize(dynamic=True),
        ]
    )

    # dense_layers_1
    dense_layers = tf.keras.models.Sequential(
        name= "dense_layers_resnet50",
        layers = [
            tf.keras.layers.GlobalAveragePooling2D(),
            tf.keras.layers.Dropout(rate=rate_i(hp,mode,0,[0.5,0.8])),
            soft_max(units=num_classes)
        ]
    )

    # compose
    model = compose_layers(preprocess_layer,pretrained_resnet50)
    model = compose_layers(model,dense_layers)

    #compile
    model.compile(loss="categorical_crossentropy",
        metrics=["accuracy"],
        optimizer=tf.keras.optimizers.RMSprop(learning_rate=learning_rate(hp,mode))
    )

    return model
```



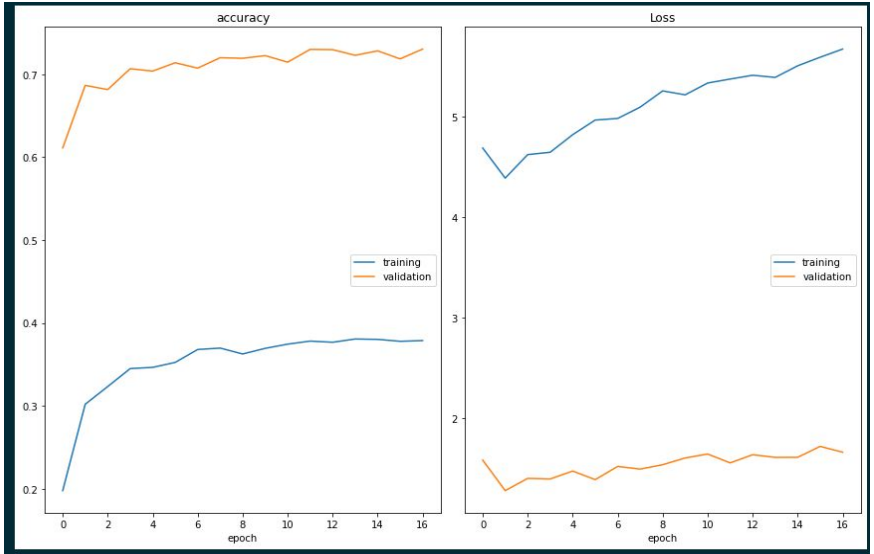
ViT

```python
def my_pretrained_vit_model(hp,mode,num_classes,
    handle="https://tfhub.dev/sayakpaul/vit_s16_fe/1"):

    hub_layer = tfhub.KerasLayer(handle,trainable=False)

    model = tf.keras.Sequential(
        name="myViT",
        layers= [
            tf.keras.layers.InputLayer(IMG_SIZE + (3,)),
            Equalize(dynamic=True),
            tf.keras.layers.Rescaling(scale=1 / 127.5, offset=-1),
            hub_layer,
            soft_max(units=num_classes)
        ]
    )

    model.compile(loss="categorical_crossentropy",
            metrics=["accuracy"],
            optimizer=tf.keras.optimizers.RMSprop(learning_rate=learning_rate(hp,mode))
          )

    return model
```



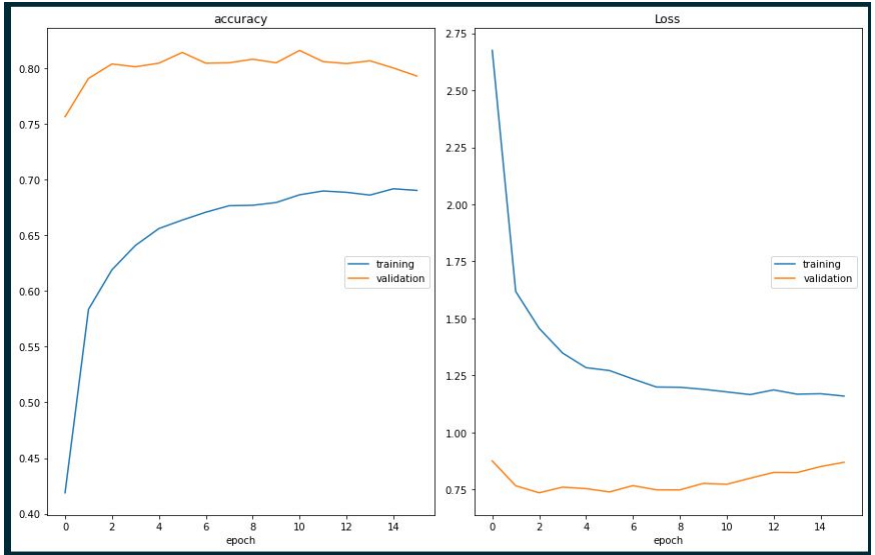
ResNet 50 Versus ViT

- ResNet 50




```
accuracy
       training                (min:    0.198, max:    0.381, cur:    0.379)
       validation              (min:    0.611, max:    0.730, cur:    0.730)
Loss
       training                (min:    4.391, max:    5.677, cur:    5.677)
       validation              (min:    1.276, max:    1.717, cur:    1.659)
Epoch 00017: early stopping
84/84 [==============================] - 49s 583ms/step - loss: 1.6407 - accuracy: 0.7078

===== EVALUATE =====

test loss, test acc: [1.64071786403656, 0.7078148126602173]

====================
```

- ViT




```
accuracy
       training                (min:    0.419, max:    0.692, cur:    0.690)
       validation              (min:    0.757, max:    0.816, cur:    0.793)
Loss
       training                (min:    1.159, max:    2.675, cur:    1.159)
       validation              (min:    0.735, max:    0.875, cur:    0.869)
Epoch 00016: early stopping
84/84 [==============================] - 55s 647ms/step - loss: 0.7562 - accuracy: 0.8084

===== EVALUATE =====

test loss, test acc: [0.75617516040802, 0.8083623647689819]

====================
```

ResNet 50 Versus ViT

Modèle	Accuracy (test)	Temps (User time GPU)
ResNet 50	0.7078	5h 6min 47s
ViT	0.8084	4h 43min 50s