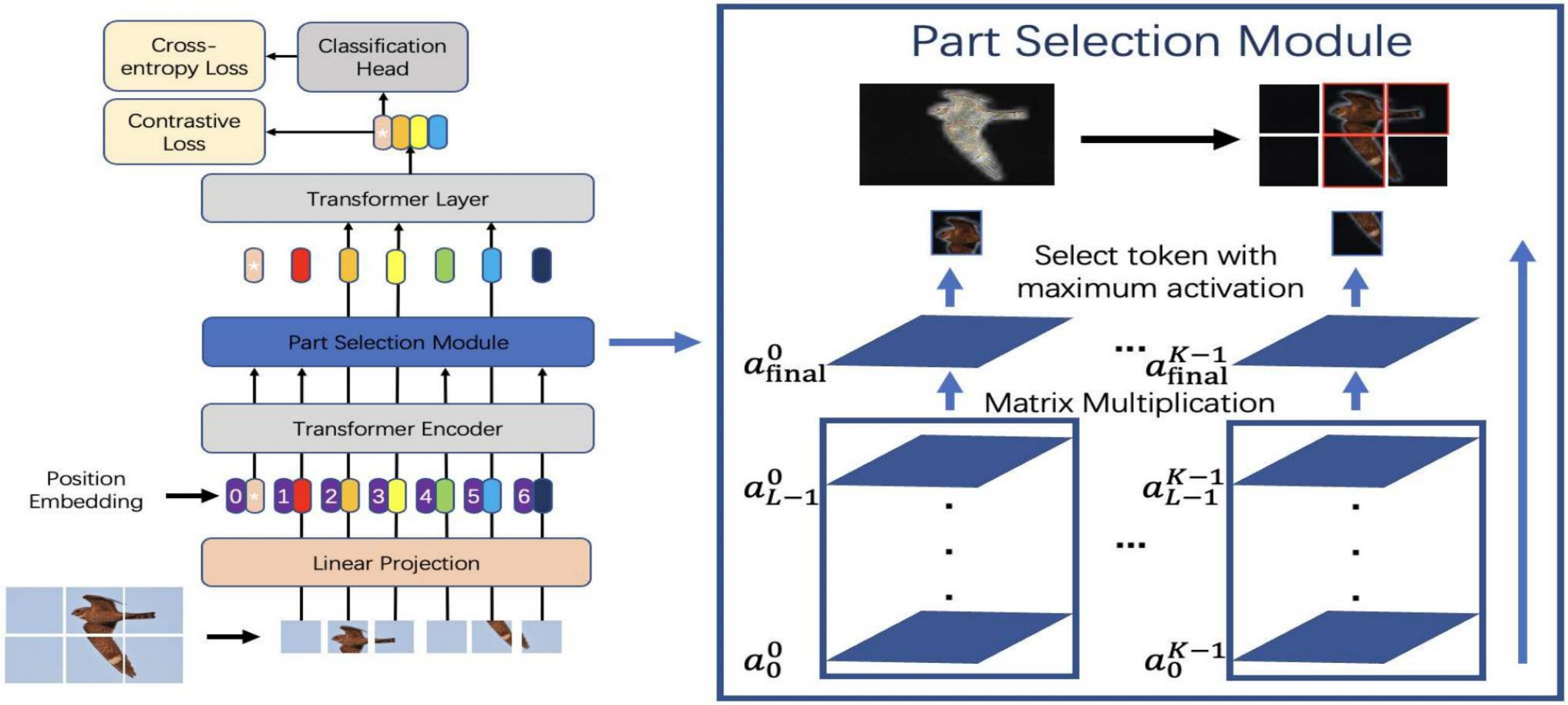
Solve FGVC based on **Vision Transformer**

Shiyu Chen, Jingyun Huang and Zijian Lin

2023.6.18

TransFG: A Transformer Architecture for Fine-Grained Recognition *--Dec 1 2021*

https://github.com/TACJu/TransFG



TransFG: A Transformer Architecture for Fine-Grained Recognition *--Dec 1 2021*

https://github.com/TACJu/TransFG

Table 1: Comparison of different methods on CUB-200-2011, Stanford Cars.

Method	Backbone	CUB	Cars
ResNet-50	ResNet-50	84.5	-
NTS-Net	ResNet-50	87.5	93.9
Cross-X	ResNet-50	87.7	94.6
DBTNet	ResNet-101	88.1	94.5
FDL	DenseNet-161	89.1	94.2
PMG	ResNet-50	89.6	95.1
API-Net	DenseNet-161	90.0	**95.3**
StackedLSTM	GoogleNet	90.4	-
DeiT	DeiT-B	90.0	93.9
ViT	ViT-B_16	90.3	93.7
TransFG	ViT-B_16	**91.7**	94.8

give qualitative analysis and visualization results to show the interpretability of our model.

Table 2: Comparison of different methods on Stanford Dogs.

Method	Backbone	Dogs
MaxEnt	DenseNet-161	83.6
FDL	DenseNet-161	84.9
Cross-X	ResNet-50	88.9
API-Net	ResNet-101	90.3
ViT	ViT-B_16	91.7
TransFG	ViT-B_16	**92.3**

Table 3: Comparison of different methods on NABirds.

Method	Backbone	NABirds
Cross-X	ResNet-50	86.4
API-Net	DenseNet-161	88.1
CS-Parts	ResNet-50	88.5
FixSENet-154	SENet-154	89.2
ViT	ViT-B_16	89.9
TransFG	ViT-B_16	**90.8**

Model structure: ViT

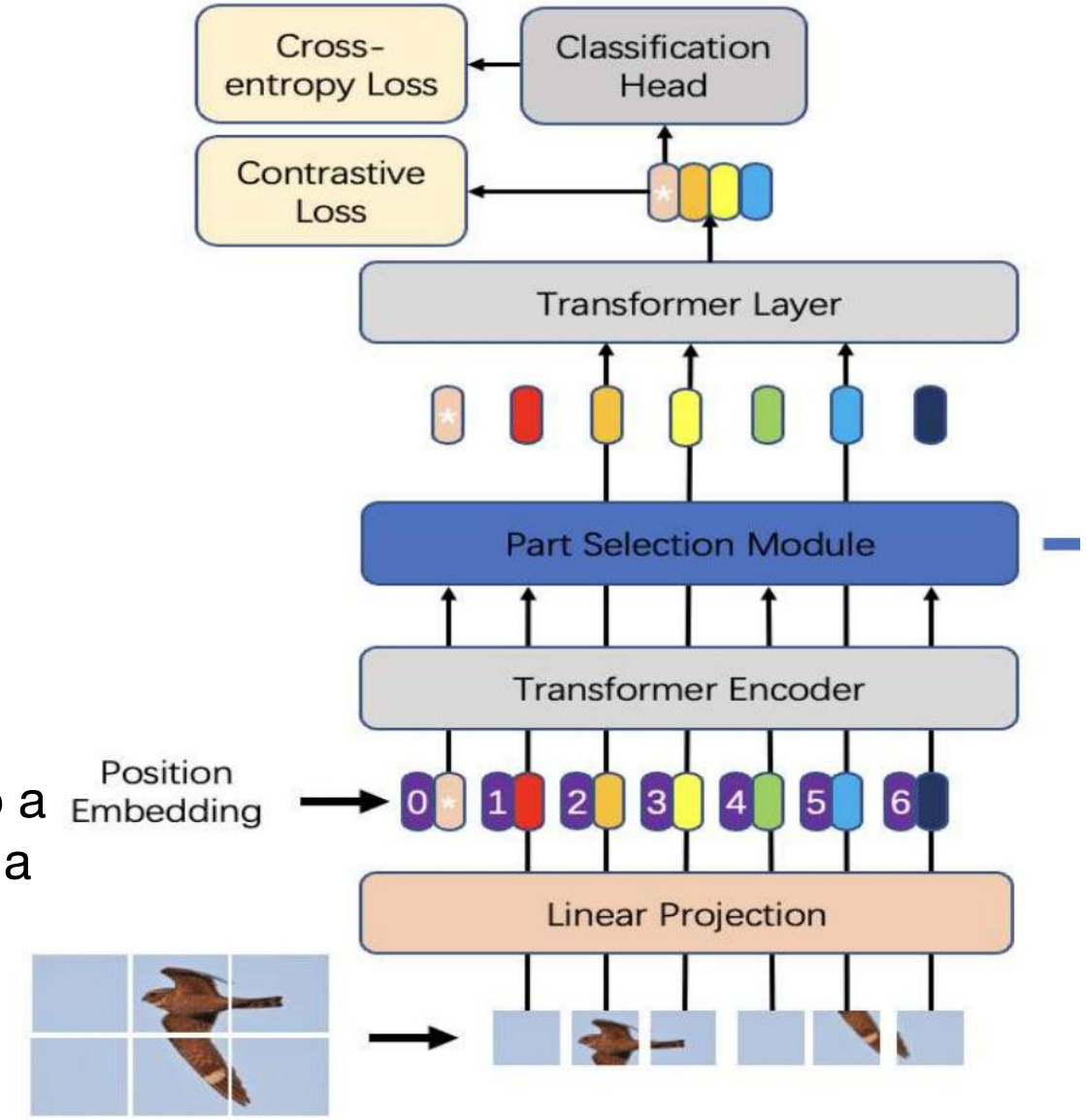
Image Sequentialization

Segmentation: Preprocessing the input image into a sequence of flattened patches.

Patch Embedding

(1)Mapping: map the vectorized patches into a latent D-dimensional embedding space using a trainable linear projection.

(2)Pass through transformer

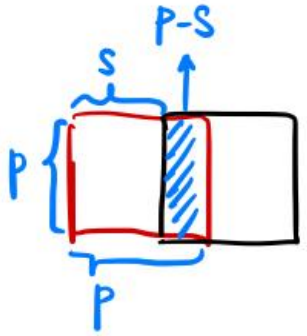


Model structure: ViT

Image Sequentialization

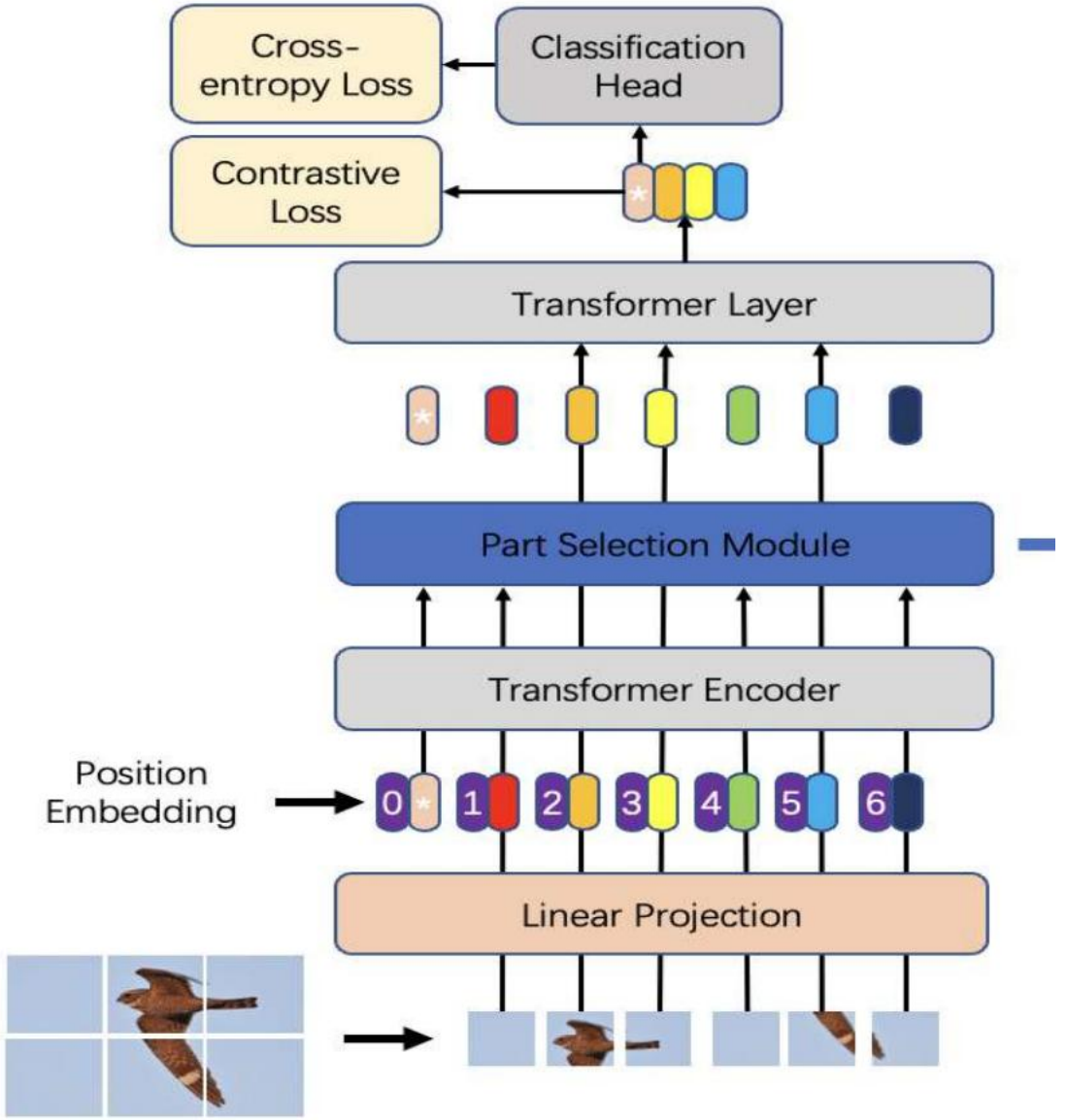
Segmentation: Preprocessing the input image into a sequence of flattened patches.

Input : H*W
Patch: P
Step: S



$$N = N_H * N_W = \lfloor \frac{H - P + S}{S} \rfloor * \lfloor \frac{W - P + S}{S} \rfloor$$

Why overlapping?



Patch Embedding

- Map patch x_p into latent D-dimensional embedding space, **E** is the patch embedding projection

$$z_0 = [x_p^1 E, x_p^2 E, ..., x_p^N E] + E_{pos}$$

- **ViT** exploits the **first token** of the **last encoder layer** z_l^0 as the representation of the global feature

$$\mathbf{z}_l^{'} = MSA(LN(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1} \qquad l \in 1, 2, \cdots, L$$

$$\mathbf{z}_l = MLP(LN(\mathbf{z}_l^{'})) + \mathbf{z}_l^{'} \qquad l \in 1, 2, \cdots, L$$

z_l is the encoded image representation

- Forward embeddings to a **classifier head** to obtain the final classification results **without considering** the potential information stored in the **rest of the tokens**

Training tricks

model	CUB200 Bird	Stanford Dogs
Ours (ViT)	90.7%	87.2%
w/o DA	89.1%	85.1%
w/o LRS	90.5%	86.8%

Table 1. The performance of some tricks. *w/o DA* means our model trains without data augmentation and *w/o LRS* means our model trains without learning rate scheduler.

model	CUB200 Bird	Stanford Dogs
w/o pretrain	88.5%	85.9%
w/ pretrain	90.7%	87.2%

Table 2. The performance of different initialization. *w/o pretrain* means training the model from scratch, and *w/ pretrain* means using the pretrained weight to initialize the model.

Object Localization

Based on R-CNN:

model	CUB200 Bird	Stanford Dogs
Baseline (Ours)	90.7%	87.2%
w/ ObLocal	90.8%	87.5%

Table 3. The performance of model with object localization. *w/ ObLocal* means that we attend object localization to our network

CNN backbone: ResNet

Backbone	CUB200 Bird	Stanford Dogs
ViT	90.7%	87.2%
ResNet	86.7%	85.8%

Table 5. The performance of model with *ResNet* backbone.

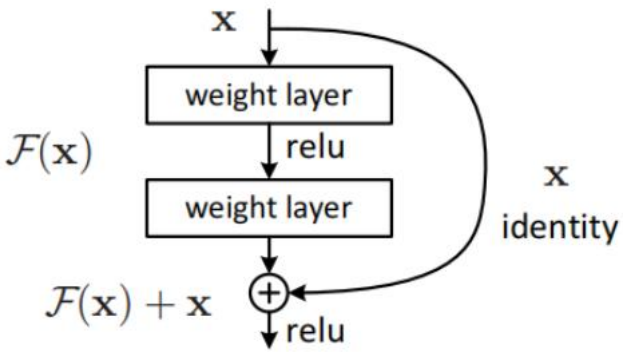


Figure 7. Structure of a residual block

ViT model backbone vs. CNN backbone

Self-attention:
ViT models use self-attention to learn relationships between different parts of the input image, while CNNs rely on local filters and convolutions. This gives ViTs the ability to capture long-range dependencies that CNNs may miss.

Translation invariance:
CNNs are translation invariant by design due to the convolutional filters. ViTs do not inherently have this property and need positional embeddings to maintain location information.

Sensitivity to input size:
ViTs generally require input images of a fixed size, while CNNs can handle a range of input sizes.