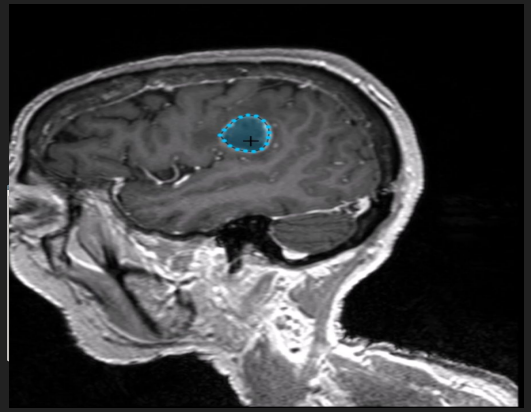
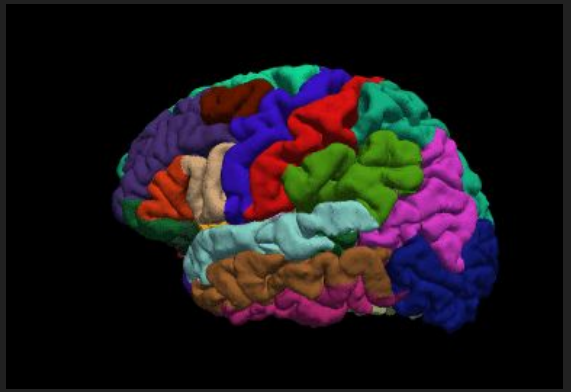
Conditional Positional Encodings for Vision Transformers

Presented by William Ashbee

Motivation

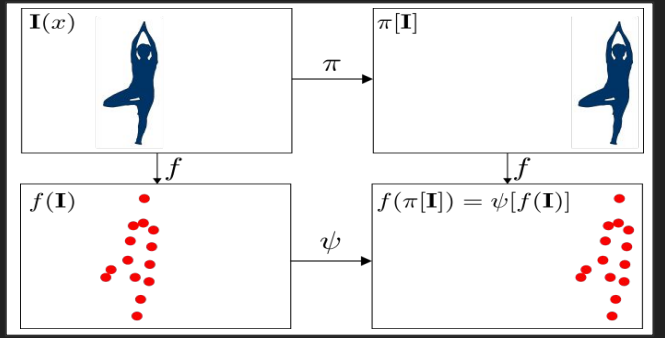
- Position has been important to most of the problems I have been working on
- Equivariance is a useful property of networks in terms of having them efficiently train and thus not waste time and resources.
- I view the ideal network as maximizing equivariance to translation, scale, rotation, etc. and maximizing capacity to be aware of position while still being fast and light weight.

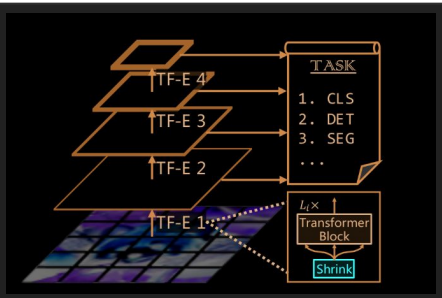


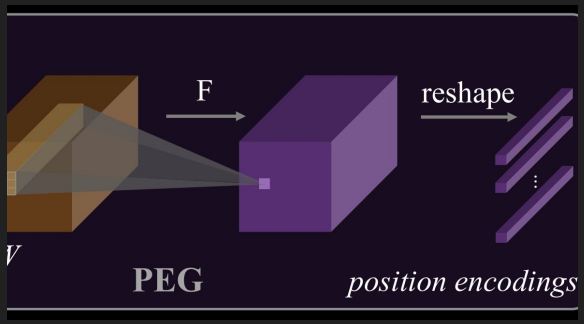


Contributions Conditional Positional Encoding (CPE)

- Promotion of Translation Equivalence: Unlike traditional absolute positional encodings, CPE provides explicit bias towards vital translation equivalence.
- Presenting CPVT: The Conditional Position Encoding Vision Transformer (CPVT) is introduced--an improvement over PVT
- Generalization Capabilities: CPE can generalize well to various input resolutions, boosting performance in segmentation and detection for pyramid transformers.

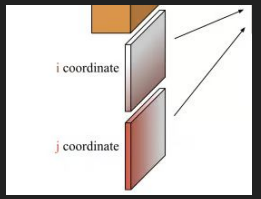


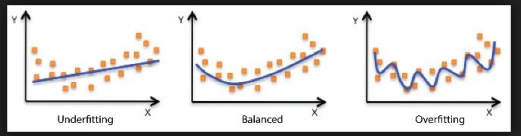


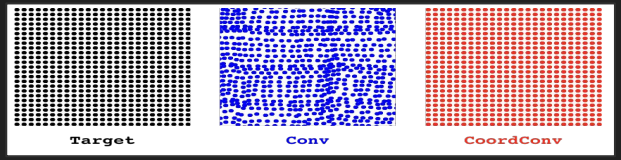


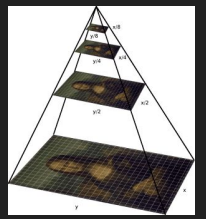
Limitations of Pre-existing Positional Encodings

- Static Nature: Traditional encodings, whether fixed or learnable, are predefined and independent of input tokens--**too top down and doesn't work well in practice**.
- Generalization Issue: Difficulty in effectively generalizing to input sequences of varying or unseen lengths during training--**not resolution equivariant for segmentation tasks**.
- Translation Equivariance: **Moving the image shouldn't cause catastrophic failures or significant data augmentation**.
- Resolution Equivariance: **Increasing resolution shouldn't require significant fine tuning on segmentation tasks**









Solutions to Limitations

- **Removing Positional Encodings**: While applicable to longer sequences, this solution significantly deteriorates model performance (e.g., DeiT-tiny's performance drops from 72.2% to 68.2% on ImageNet without positional encodings).
- **Interpolating Position Encodings**: Despite being a plausible solution to accommodate longer sequences, this requires additional model fine-tuning and doesn't inherently enhance performance with higher-resolution inputs.
- **Create newer, better encodings**.

Position Encoding Generator (PEG)

- Purpose: Implemented to facilitate the incorporation of CPE into the Transformer framework--**modularity**.
- Functionality: A simple generator that allows CPE to be effortlessly merged into existing systems--**easy integration**.

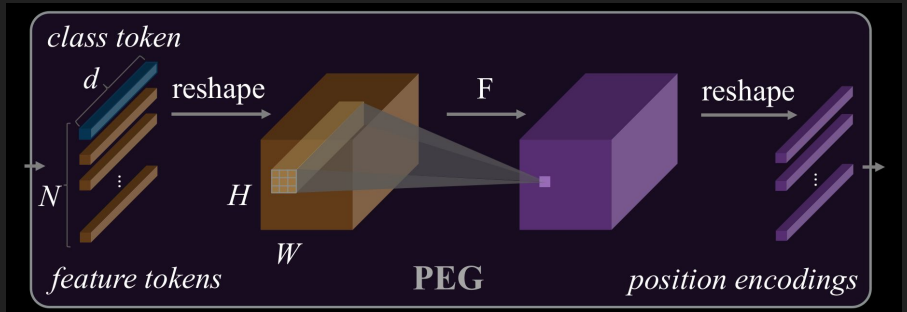


Figure 2. Schematic illustration of Positional Encoding Generator (PEG). Note d is the embedding size, N is the number of tokens.

Conditional Positional Encoding Generators (PEG)

- Mechanism: PEG dynamically generates positional encodings, conditioned on an input token's local neighborhood--**basically convolutions**.
- Local Relationship Characterization: **Sufficient to meet all key requirements stated previously**. A token's local relationships offer **permutation-variability, translational-equivariance, and generalizability to higher resolutions**.
- Zero Padding Significance: Essential for enabling the model to comprehend absolute positions--**Absolute position can be learned**.
- PEG **reshapes** the flattened input sequence, utilizes **function F**, and generates conditional positional encodings.

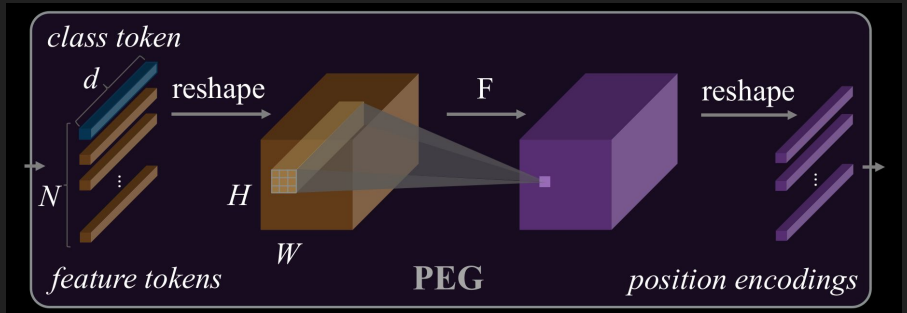
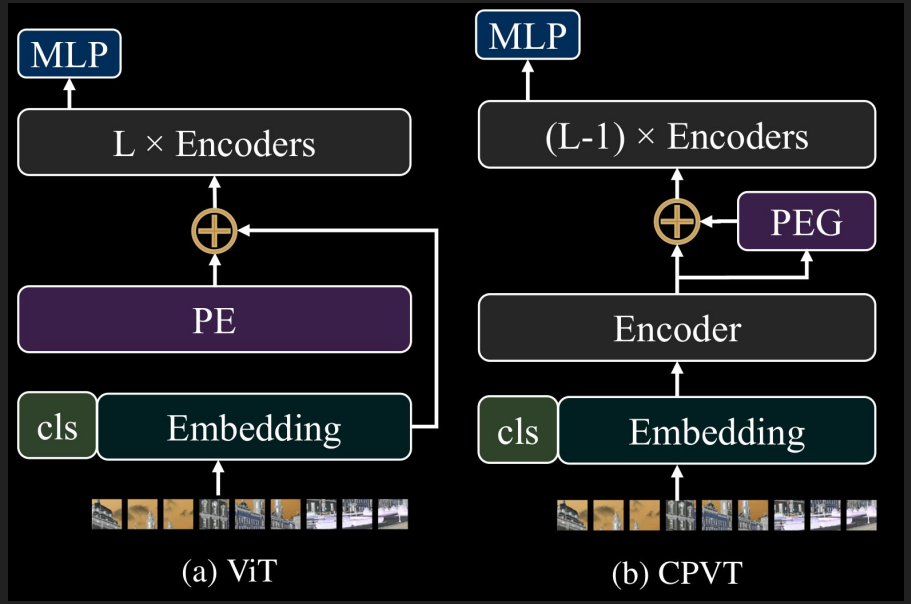
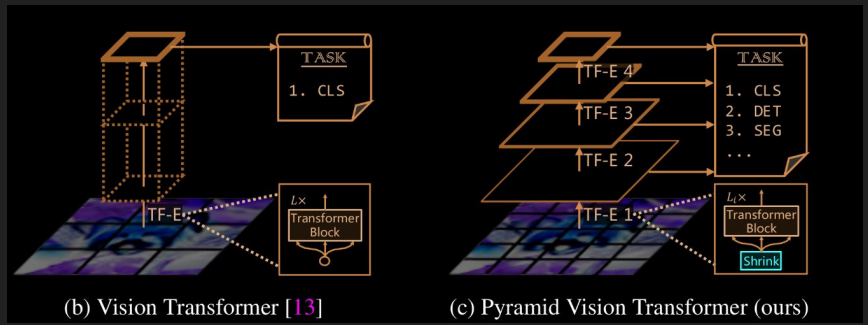


Figure 2. Schematic illustration of Positional Encoding Generator (PEG). Note d is the embedding size, N is the number of tokens.

The CPVT Model

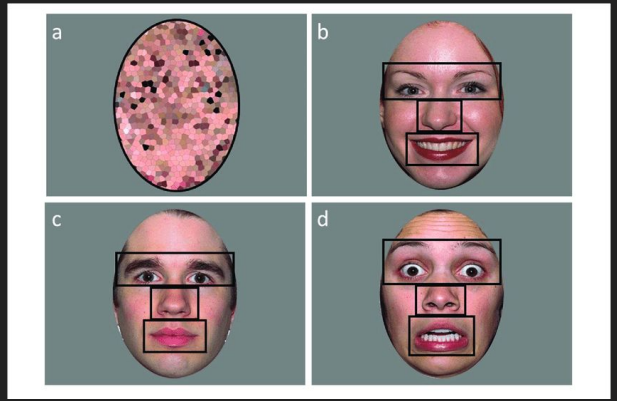
- CPVT: A model developed upon PEG, dedicated to seamlessly incorporating CPE into Vision Transformers.
  - Image pyramid (Pyramid Vision Transformer)
  - Conditional Positional Encoding (CNN)
- Performance: Showcases superior results and outperforming benchmarks in pertinent evaluations.

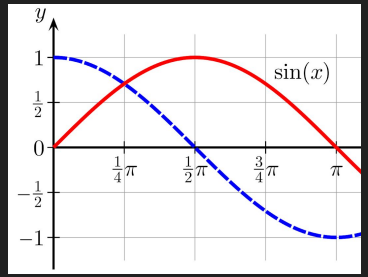


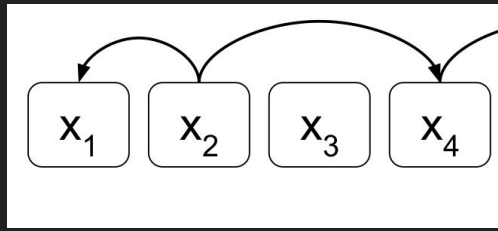


Related Work in Positional Encoding

- The Need for Positional Encoding: Given that self-attention is permutation-equivariant, positional encodings play a crucial role in embedding the order of sequences.
- Absolute vs. Relative Encodings: Positional encodings can either be absolute or relative and can be fixed (such as sinusoidal functions) or learnable (updated during training).
- **Classification could be done without position, but segmentation cannot be.**

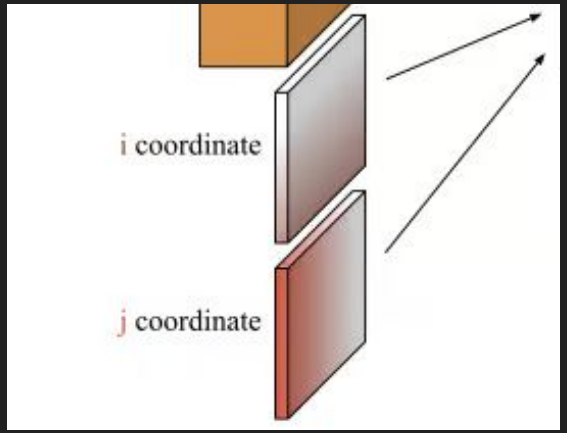


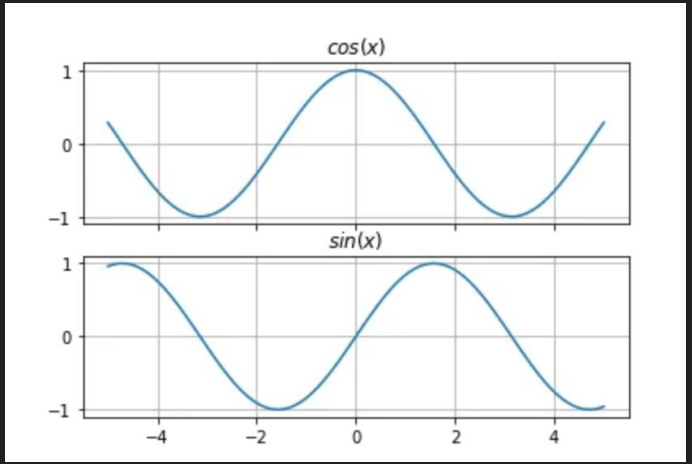




Absolute Positional Encoding

- Widely Adopted Method: Absolute positional encoding is commonly utilized in the original transformer, using either sinusoidal functions or learnable encodings.
- Mechanism: Sinusoidal functions generate encodings of different frequencies, or alternatively, a fixed-dimension matrix/tensor for learnable encodings is jointly updated with model parameters.





Relative Positional Encoding and Its Developments

- **Considering Token Distances**: Relative positional encoding considers the distances between tokens in the input sequence, providing translation-equivariance and natural handling of longer sequences.
- **2-D Relative Position Encoding**: This approach **for image classification shows superiority over 2D sinusoidal embeddings**.
- **Addressing Resolution and Translation Equivariance**: Relative position encodings seem capable of handling different resolutions, etc..

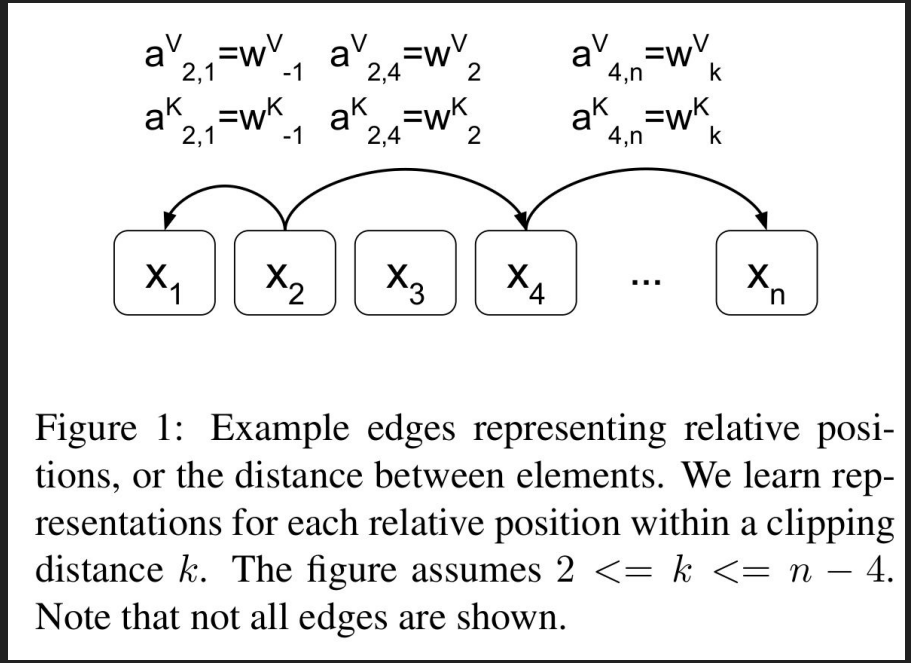
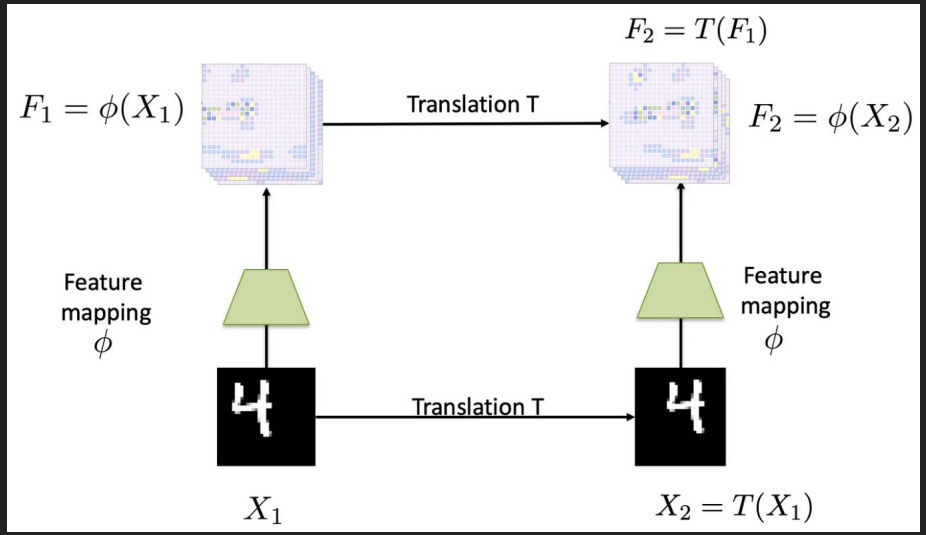


Figure 1: Example edges representing relative positions, or the distance between elements. We learn representations for each relative position within a clipping distance k. The figure assumes $2 <= k <= n - 4$. Note that not all edges are shown.

In search of Translational Equivariance in Transformers

- The conventional positional encodings present limitations related to handling longer sequences and ensuring translation-equivariance.
- Translation equivariance is particularly useful in segmentation because when the MRI shifts, so will the segmentation.
- Altered positional encodings better preserve this property



Ablation - Traditional Positional Encoding Strategies on ImageNet

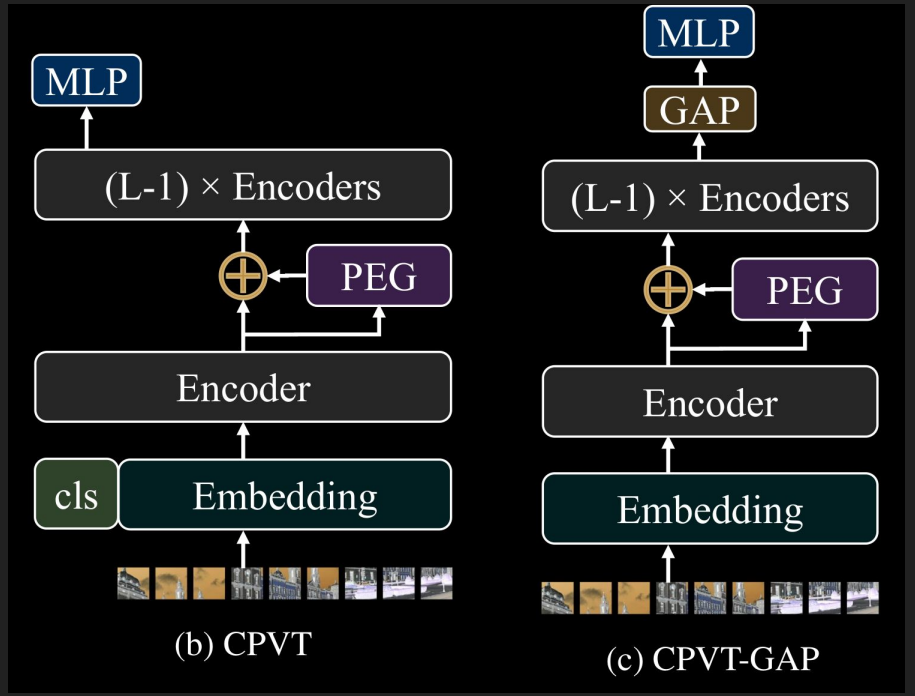
Table 1. Comparison of various positional encoding (PE) strategies tested on ImageNet validation set in terms of the top-1 accuracy. Removing the positional encodings greatly damages the performance. The relative positional encodings have inferior performance to the absolute ones

Model	Encoding	Top-1@224(%)	Top-1@384(%)
DeiT-tiny (Touvron et al., 2020)	✗	68.2	68.6
DeiT-tiny (Touvron et al., 2020)	learnable	72.2	71.2
DeiT-tiny (Touvron et al., 2020)	sin-cos	72.3	70.8
DeiT-tiny	2D RPE (Shaw et al., 2018)	70.5	69.8

- Eliminating PEs severely compromises performance.
- Relative PEs have suboptimal performance compared to absolute PEs, even though they can address certain challenges. For instance, a performance drop from 72.2% to 70.5% using 2D RPE.

Employing Distinct Strategies for Classification in CPVT

- **Classification Tokens in ViT & DeiT**: Both models leverage an extra learnable class token (cls token) to perform classification, which by design isn't translation-invariant,
- Inherent Translation Invariance with GAP--**Global Average Pooling**: Introducing CPVT-GAP, where a global average pooling (GAP) replaces the class token, providing an inherently translation-invariant solution and subsequently enhancing image classification performance.



Comparison - DeiT V CPVT - Generalization to Higher Resolutions

Table 2. Direct evaluation on other resolutions without fine-tuning. The models are trained on 224×224. A simple PEG of a single layer of 3×3 depth-wise convolution is used here

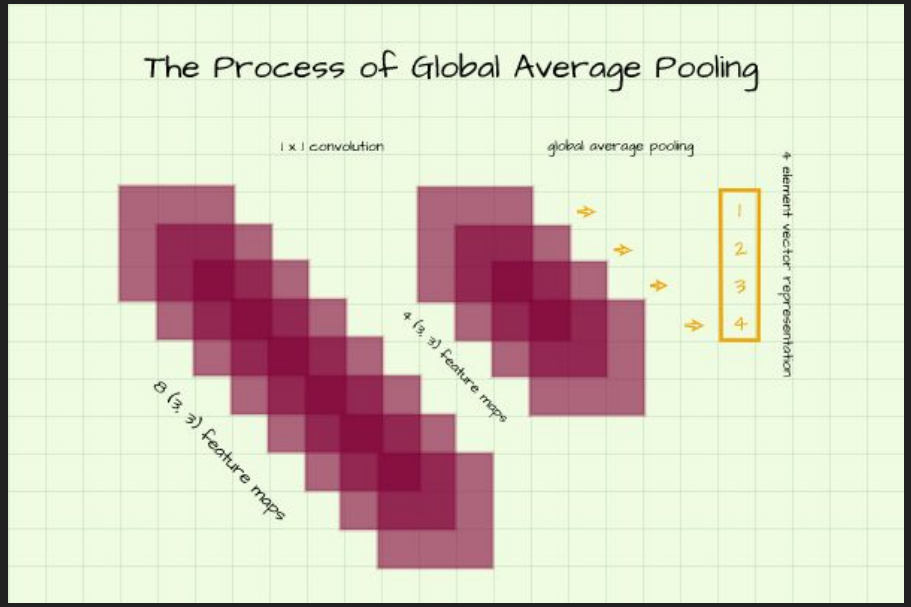
Model	Params	160(%)	224(%)	384(%)	448(%)	512(%)
DeiT-tiny	6M	65.6	72.2	71.2	68.8	65.9
DeiT-tiny (sin)	6M	65.2	72.3	70.8	68.2	65.1
DeiT-tiny (no pos)	6M	62.1	68.2	68.6	68.4	65.0
CPVT-Ti	6M	66.8(+1.2)	72.4(+0.2)	73.2(+2.0)	71.8(+3.0)	70.3(+4.4)
CPVT-Ti ‡	6M	67.7 (+2.1)	73.4(+1.2)	74.2(+3.0)	72.6(+3.8)	70.8(+4.9)
DeiT-small	22M	75.6	79.9	78.1	75.9	72.6
CPVT-S	22M	76.1(+0.5)	79.9	80.4(+1.5)	78.6(+2.7)	76.8(+4.2)
DeiT-base	86M	79.1	81.8	79.7	79.8	78.2
CPVT-B	86M	80.5(+1.4)	81.9(+0.1)	82.3(+2.6)	82.4(+2.6)	81.0(+2.8)

‡: Insert one PEG each after the first encoder till the fifth encoder

- **No Fine-Tuning Required**: Highlighting PEG's ability to effortlessly scale to larger image sizes.
- Comparative Insight: Through direct evaluation on multiple resolutions, CPVT showcases its prowess by boosting performance (CPVT-Ti: 73.4% to 74.2% on 384 × 384 images), offering a significant edge over DeiT-tiny.
- - models trained on 224×224 and directly evaluated on 384×384, 448×448, and 512×512.

Enhancing CPVT with Global Average Pooling

- Positional Encoding: **CPVT harnesses PEG**, which is inherently translation-equivariant.
- **Leveraging GAP**: Substituting the cls token with Global Average Pooling (GAP), an inherently translation-invariant method, optimizes computation and elevates performance.
- **Performance Gains**: Utilizing GAP facilitates over a 1% boost in CPVT, outpacing DeiT-tiny notably (+2.7% with CPVT-Ti).



Deep Dive into GAP's Impact

Table 3. Performance comparison of Class Token (CLT) and global average pooling (GAP) on ImageNet. CPVT's can be further boosted with GAP

Model	Head	Params	Top-1 Acc (%)	Top-5 Acc (%)
DeiT-tiny (Touvron et al., 2020)	CLT	6M	72.2	91.0
DeiT-tiny	GAP	6M	72.6	91.2
CPVT-Ti ‡	CLT	6M	73.4	91.8
CPVT-Ti ‡	GAP	6M	**74.9**	**92.6**
DeiT-small (Touvron et al., 2020)	CLT	22M	79.9	95.0
DeiT-small	GAP	22M	80.2	95.2
CPVT-S ‡	CLT	23M	80.5	95.2
CPVT-S ‡	GAP	23M	**81.5**	**95.7**

‡: Insert one PEG each after the first encoder till the fifth encoder

- **Intriguing Discrepancies**: DeiT with GAP does not witness significant improvements (only +0.4%) due to its original non-translation-equivariant PE.
- **Superiority of CPVT**: CPVT-Ti with GAP attains 74.9% top-1 accuracy on the ImageNet validation dataset, eclipsing not only DeiT-tiny but also **its distilled variant**.

Performance Metrics

- CPVT models demonstrate appreciably superior top-1 accuracy compared to DeiT variants.
- Utilizing GAP enables CPVT to establish a new pinnacle of performance amongst vision Transformers.

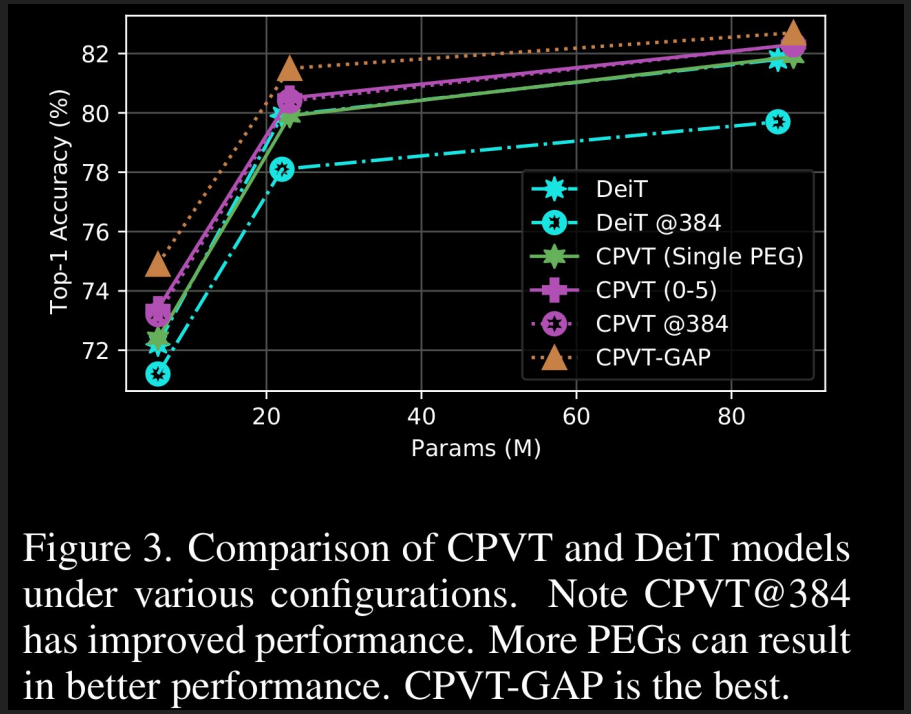


Figure 3. Comparison of CPVT and DeiT models under various configurations. Note CPVT@384 has improved performance. More PEGs can result in better performance. CPVT-GAP is the best.

Ablation - PEG function types

- Observe that a random initialized convolution is better than no positional encoding at all
- A sized 3, learned kernel is best.

Table 5. Positional encoding rather than added parameters gives the most improvement

Kernel	Style	Params (M)	Top-1 Acc (%)
none	-	5.68	68.2
3	fixed (random init)	5.68	71.3
3	fixed (learned init)	5.68	72.3
1 (12 $\times$)	learnable	6.13	68.6
3	learnable	5.68	**72.4**

Ablation - different PEG layer

Table 6. Comparison of different plugin positions (left) and kernels (right) using DeiT-tiny

PosIdx	Top-1 (%)	Top-5 (%)
none	68.2	88.7
−1	70.6	90.2
0	**72.4**	**91.2**
3	72.3	91.1
6	71.7	90.8
10	69.0	89.1

PosIdx	kernel	Params	Top-1 (%)	Top-5 (%)
-1	3×3	5.7M	70.6	90.2
-1	27×27	5.8M	**72.5**	**91.3**

- Positional encodings placed at different layers produce different results.

Comparison - CPVT vs DeiT

Table 7. Comparison of various positional encoding strategies. LE: learnable positional encoding. RPE: relative positional encoding

Model	PEG Pos	Encoding	Top-1 (%)	Top-5 (%)
DeiT-tiny (2020)	-	LE	72.2	91.0
DeiT-tiny	-	2D sin-cos	72.3	91.0
DeiT-tiny	-	2D RPE	70.5	90.0
CPVT-Ti	0-1	PEG	72.4	91.2
CPVT-Ti	0-1	PEG + LE	72.9	91.4
CPVT-Ti	0-1	4×PEG + LE	72.9	91.4
CPVT-Ti	0-5	PEG	**73.4**	**91.8**

- CPVT with Peg is able to beat DeiT with traditional positional encodings

Ablation - DeiT with traditional vs DeiT with PEG

Table 8. Direct evaluation on other resolutions without fine-tuning. The models are trained on 224×224. CPE outperforms LE+RPE combination on untrained resolutions.

Model	Positional Params	160(%)	224(%)	384(%)	448(%)	512(%)
DeiT-tiny (LE+RPE)	40011	65.6	72.4	70.8	68.4	65.6
DeiT-tiny (PEG at Pos 0)	1920	66.8	72.4	73.2	71.8	70.3

- Demonstrating improved generalizability of PEG against learned and relative positional encodings.
- Fewer parameters generalize better to higher resolutions.

Hyperparameters used during training

- Hyperparameters used during training
- Of interest is that dropout is replaced with stochastic depth
- Gradient clipping is removed
- Other advanced methods added to newer generation

Table 10. Hyper-parameters for ViT, DeiT and CPVT

Methods	ViT	DeiT	CPVT
Epochs	300	300	300
Batch size	4096	1024	1024
Optimizer	AdamW	AdamW	LAMB
Learning rate decay	cosine	cosine	cosine
Weight decay	0.3	0.05	0.05
Warmup epochs	3.4	5	5
Label smoothing ε (Szegedy et al., 2016)	✗	0.1	0.1
Dropout (Srivastava et al., 2014)	0.1	✗	✗
Stoch. Depth (Huang et al., 2016)	✗	0.1	0.1
Repeated Aug (Hoffer et al., 2020)	✗	✓	✓
Gradient Clip.	✓	✗	✗
Rand Augment (Cubuk et al., 2020)	✗	9/0.5	9/0.5
Mixup prob. (Zhang et al., 2018)	✗	0.8	0.8
Cutmix prob. (Yun et al., 2019)	✗	1.0	1.0
Erasing prob. (Zhong et al., 2020)	✗	0.25	0.25

Ablation - Importance of zero paddings with PEG

Table 11. Ablation study on ImageNet performance w/ or w/o zero paddings

Model	Padding	Top-1 Acc(%)	Top-5 Acc(%)
CPVT-Ti	✓	**72.4**	**91.2**
	✗	70.5	89.8

- Recall imagenet class is usually in the center.
- Absolute position encodings help determine location of the center.
- Zero padding with Peg gives absolute position information.

Ablation - Multi Level PEG

B.4 SINGLE PEG VS. MULTIPLE PEGS

We further evaluate whether or not using *multi-position* encodings can benefit the performance in Table 12. Notice we denote by i-j the inserted positions of PEG which start from the i-th encoder and end at the $j-1$-th one (inclusion). By inserting PEGs to five positions, the top-1 accuracy of the tiny model can achieve 73.4%, which surpasses DeiT-tiny by 1.2%. Similarly, CPVT-S can achieve 80.5%. It turns out more PEGs do help, but up to a level where more PEGs become incremental (0-5 vs. 0-11).

Table 12. CPVT's sensitivity to number of plugin positions

Positions	Model	Params (M)	Top-1 Acc (%)	Top-5 Acc (%)
0-1	tiny	5.7	72.4	91.2
0-5	tiny	5.9	**73.4**	**91.8**
0-11	tiny	6.1	**73.4**	**91.8**
0-1	small	22.0	79.9	95.0
0-5	small	22.9	80.5	**95.2**
0-11	small	23.8	**80.6**	**95.2**

Ablation - PVT

Table 13. Our method boosts the performance of PVT on ImageNet classification, ADE20K segmentation and COCO detection

Backbone	ImageNet		Semantic FPN on ADE20K		RetinaNet on COCO		
	Params (M)	Top-1 (%)	Params (M)	mIoU (%)	Params (M)	mAP (%, 1×)	mAP (%, 3×, +MS)
ResNet-18 (He et al., 2016)	12	69.8	16	32.9	21	31.8	35.4
PVT-tiny (Wang et al., 2021)	13	75.0	17	35.7	23	36.7	39.4
PVT-tiny+PEG	13	77.3	17	38.0	23	38.0	**41.8**
PVT-tiny+GAP	13	75.9	17	36.0	23	36.9	39.7
PVT-tiny+PEG+GAP	13	**78.1**	17	**38.8**	23	**38.7**	**41.8**
PVT-small (Wang et al., 2021)	25	79.8	28	39.8	34	40.4	42.2
PVT-small+PEG+GAP	25	**81.2**	28	**44.3**	34	**43.0**	**45.2**
PVT-Medium (Wang et al., 2021)	44	81.2	48	41.6	54	41.9	43.2
PVT-Medium+PEG+GAP	44	**82.7**	48	**44.9**	54	**44.3**	**46.4**

An mIoU score above 0.42 on ADE20K is considered respectable given the complexity and diversity of the dataset.

Scores in the range of 0.50 and above could be considered very competitive or state-of-the-art, depending on the specific benchmarks and model variants.

B.7 Ablation on Other Forms of PEG

We explore several forms of PEG based on the tiny model, which change the type of convolution, kernel size and layers. The inserted position is 0. The result is shown in Table 14. When we use large kernel of 7×7 or dense convolution, the performance improvement is limited. Stacking more layers of depth-wise convolution doesn't bring significant improvement. Therefore, we use the simplest form as our default implementation. It indicates that this design is enough to provide good position information.

Table 14. Other forms of PEG. The simple form of a single depth-wise 3×3 is good enough.

Variants	Model	Top-1 Acc (%)
1 Depthwise Conv 3×3	tiny	72.4
1 Depthwise Conv 7×7	tiny	72.5
4 * (Depthwise Conv 3×3 +BN+ReLU)	tiny	72.4
1 Dense Conv 3×3	tiny	72.3
4 * (Dense Conv 3×3+BN+ReLU)	tiny	72.5

Simple functions work well enough.

Algorithm 1 PyTorch snippet of PEG.

```
import torch
import torch.nn as nn
class VisionTransformer:
  def __init__(layers=12, dim=192, nhead=3, img_size=224, patch_size=16):
    self.pos_block = PEG(dim)
    self.blocks = nn.ModuleList([TransformerEncoderLayer(dim, nhead, dim*4) for _ in range(
        layers)])
    self.patch_embed = PatchEmbed(img_size, patch_size, dim*4)
  def forward_features(self, x):
    B, C, H, W = x.shape
    x, patch_size = self.patch_embed(x)
    _H, _W = H // patch_size, W // patch_size
    x = torch.cat((self.cls_tokens, x), dim=1)
    for i, blk in enumerate(self.blocks):
      x = blk(x)
      if i == 0:
        x = self.pos_block(x, _H, _W)
    return x[:, 0]


class PEG(nn.Module):
  def __init__(self, dim=2\textsc{56}, k=3):
    self.pos = nn.Conv2d(dim, dim, k, 1, k//2, groups=dim) # Only for demo use, more
        complicated functions are effective too.
  def forward(self, x, H, W):
    B, N, C = x.shape
    cls_token, feat_tokens = x[:, 0], x[:, 1:]
    feat_tokens = feat_tokens.transpose(1, 2).view(B, C, H, W)
    x = self.pos(feat_tokens) + feat_tokens
    x = x.flatten(2).transpose(1, 2)
    x = torch.cat((cls_token.unsqueeze(1), x), dim=1)
    return x
```

Attention maps - DeiT w/o PE, DeiT, CPVT

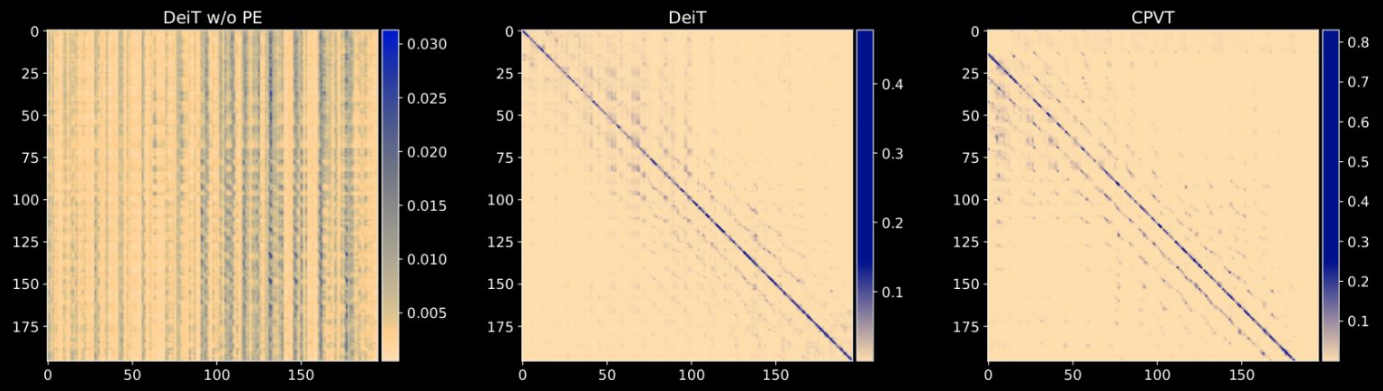


Figure 5. Normalized attention scores (first head) of the second encoder block of DeiT without position encoding (DeiT w/o PE), DeiT (Touvron et al., 2020), and CPVT on the same input sequence. Position encodings are key to developing a schema of locality in lower layers of DeiT. Meantime, CPVT profits from conditional encodings and follows a similar locality pattern.

Positional encodings dramatically change the attention maps

Attention Maps - DeiT v CPVT; 2nd & 3rd Head, 2nd & 3rd encoder

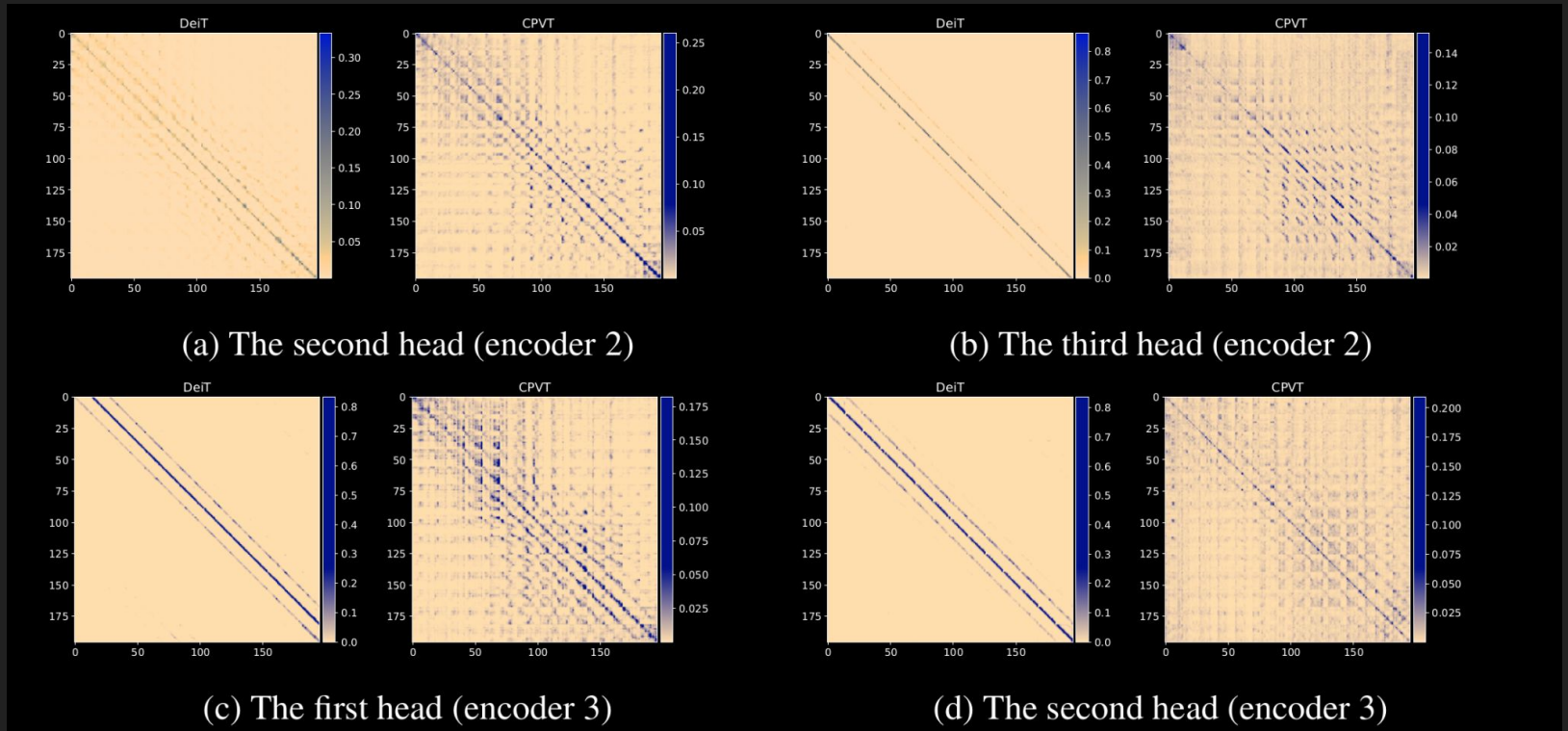
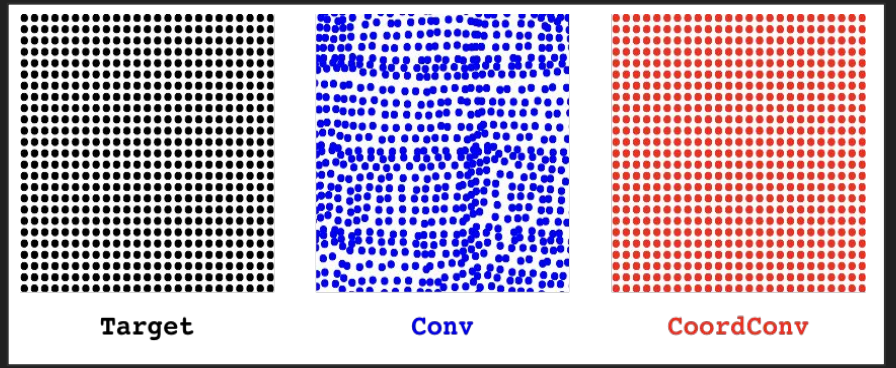
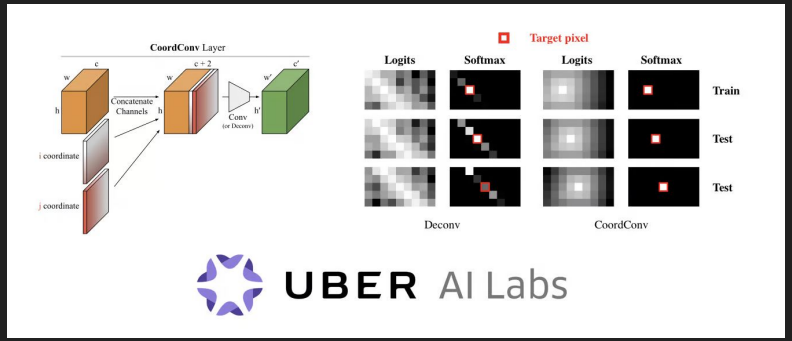


Figure 6. Normalized attention scores (the second and third head) of the second and third encoder block of DeiT (Touvron et al., 2020), and CPVT on the same input sequence. DeiT and CPVT share similar locality patterns that are aligned diagonally (some might shift).

My thoughts on positional encoding

- The way we add **Positional encoding in at a later stage** is useful **numerical hack similar** to a learned or precomputed bias, or the hack performed in residual connections, but **decreases interpretability for the sake of computational efficiency**.
- Positional encoding could easily be just **an additional channel for the input of each layer**, although it might take a few iterations on this concept to make it as efficient as current methods. And I see no reason to limit the positional encoding to a single method (channel) on a new problem except memory optimization.
- **Absolute position, relative position, and conditional (neighborhood) position all seem to have value in different types of problems.**
- Positional encoding is providing enough context to the element that will become permutation invariant in order for the permutation invariance not to matter. E.g. (**channel logic**) adding information to a pixel, voxel, or patch that preserves the inductive bias of position even though self attention is permutation equivariant.
- **Uber published a paper on using channels for position on cnns**.





This would be an absolute positional encoding for CNN.

Thank you!


thank you
danke
謝謝
ngiyabonga
شكراً جزيلاً
рахмат
teşekkür ederim
Баярлалаа
спасибо
gracias
dank je
tapadh leat
faafetai lava
vinaka
спасибі
blagodaram
misaotra
matondo
paldies
grazzi
mahalo
хвала
asante
manana
obrigada
tenki
kiitos
dankie
nanni
nandri
dhanyavad
gràcie
hvala
mauruuru
köszönöm
akun
dankon
aciu
bedankt
bayarlalaa
enkosi
djiere dieuf
tau
mochchakkeram
дякую
mamnun
chakrane
murakoze
dziękuję
chnorakaloutioun
gratias ago
gràcies
sulpây
go raibh maith agat
sobodi
dèkuji
obrigado
sagolun
sukriya
kop khun krap
taiku
arigatô
takk
dakujem
trugarez
mèsi
najis tuke
ありがとう
tanemirt
grazie
didi madloba
kam sah hamnida
rahmat
terima kasih
rahmet
diolch
dhanyavadagalu
shukriya
merci
мерси
তোমাকে ধন্যবাদ
감사합니다
xièxie
ευχαριστώ