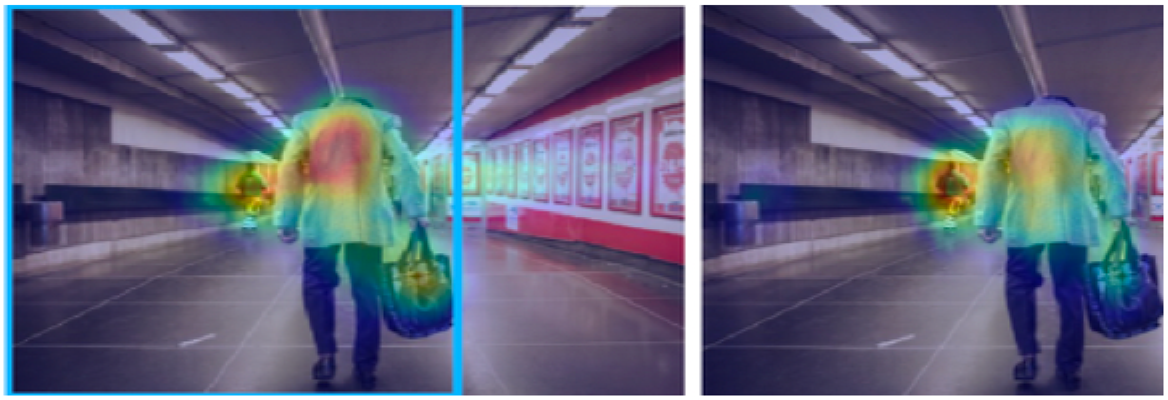
# How much positional information can a Convolutional Neural Network Encode? (ICLR 2020)

Convolutional neural networks (CNN) have achieved state of the art performance on many computer vision tasks like classification, detection, semantic segmentation etc., but has faced some criticism in context of deep learning due to the lack of interpretability.

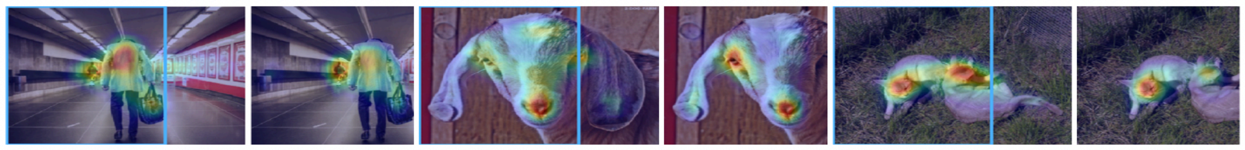
CNN are considered spatially agnostic, but it is unclear if they capture any absolute spatial information that is important for position dependent task like semantic segmentation.

In saliency detection task, in the example below,



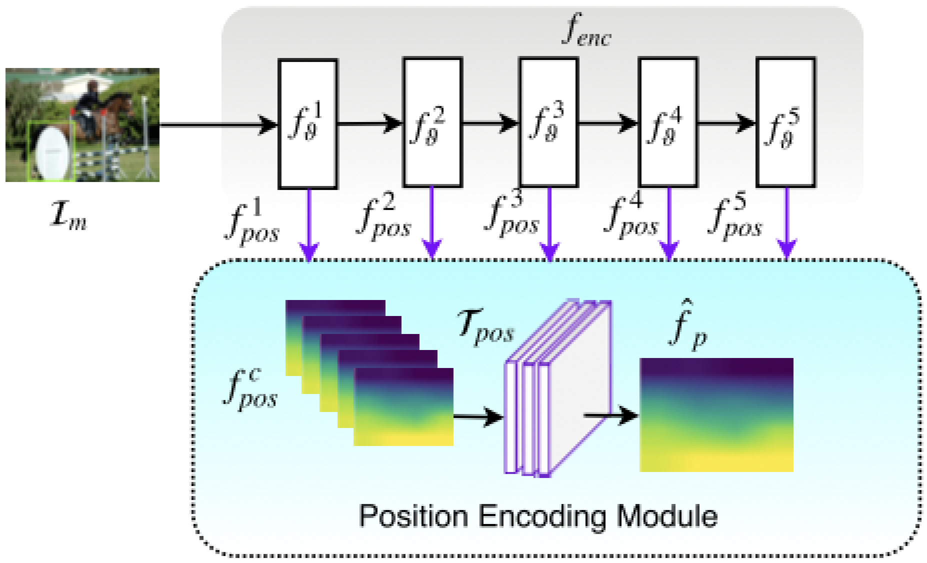
left - original image overlayed with saliency map     right- cropped image saliency map

we can see that, the position of the most salient region is shifted when the image is cropped although the features in the image remain the same. This leads us to think, if CNN encodes positional information that makes it to look at the same features again, but deem different regions as salient features with change in their position. There are few more examples from the paper below.



So, the paper hypothesize that the positional information is innately learned by the CNN and use it as a cue for decision making. They develop a network posENet and perform experiments to support their hypothesis.

The idea behind the posENet is, the positional information is implicitly encoded in the  feature maps and play prominent role in tasks like classification, semantic segmentation, saliency detection tasks etc.,



posENet architecture

## posENet architecture

posENet architecture consists of 2 parts. An encoder part (f enc) from which feature maps are extracted and a Position Encoding module (Pem)

**Encoder :** vgg/Resnet based networks after removing average pooling and classification layers is used as encoders.

The encoder network in divided into 5 feature extraction blocks, and the features are extracted from the blocks. After extraction, the bi-linear interpolation is applied on all the feature maps to bring them to same spatial dimension, and then they are concatenated.

After which these feature maps are passed into a transformation function which applies a CNN (3×3) to it and generates the output feature map. The operation is described below: W pos are trainable weights.

$$f_{pos}^c = (f_{pos}^1 \oplus \cdots \oplus f_{pos}^5) \quad \hat{f}_p = (\mathbf{W_{pos}^c} * f_{pos}^c)$$

The goal of the posENet is to predict a gradient-like position information map where each pixel defines the absolute coordinates of a pixel in an image from left to right and top to bottom.

**Ground truth?**

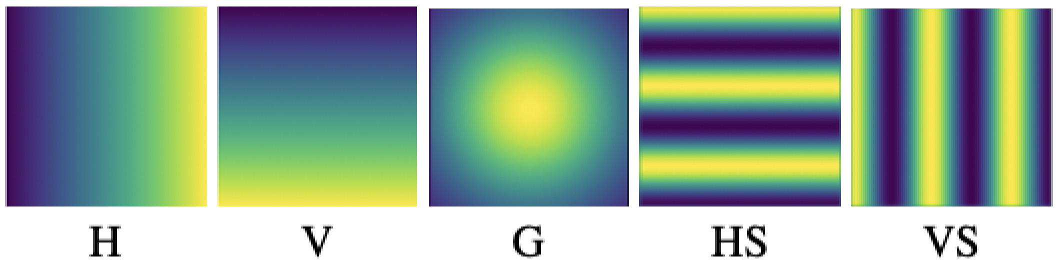


Figure 3: Sample images and generated gradient-like ground-truth position maps.

Gradient like position masks in horizontal direction (**H**), vertical direction (**V**) and Gaussian distribution map (**G**)**.** The idea of these ground truths is to see if the network can learn absolute position on one or more axes.

We can observe that, for an input natural image, there is going to be no correlation with the ground truth, and the ground truth can be thought of an random label

## Training

Load the encoder network with pre-trained imagenet weights, and freeze the network. The position encoding module generates the position maps. Pixel-wise MSE is used as the loss function to measure difference between the predicted position maps and ground-truth position maps.

$$\Delta_{\hat{f}_p} = \frac{1}{2n} \sum_{i=1}^{n} (x_i - y_i)^2$$

$x_i$ and $y_i$ refer to a pixel of $\hat{f}_p$ and $\mathcal{G}^h_{pos}$ respectively.

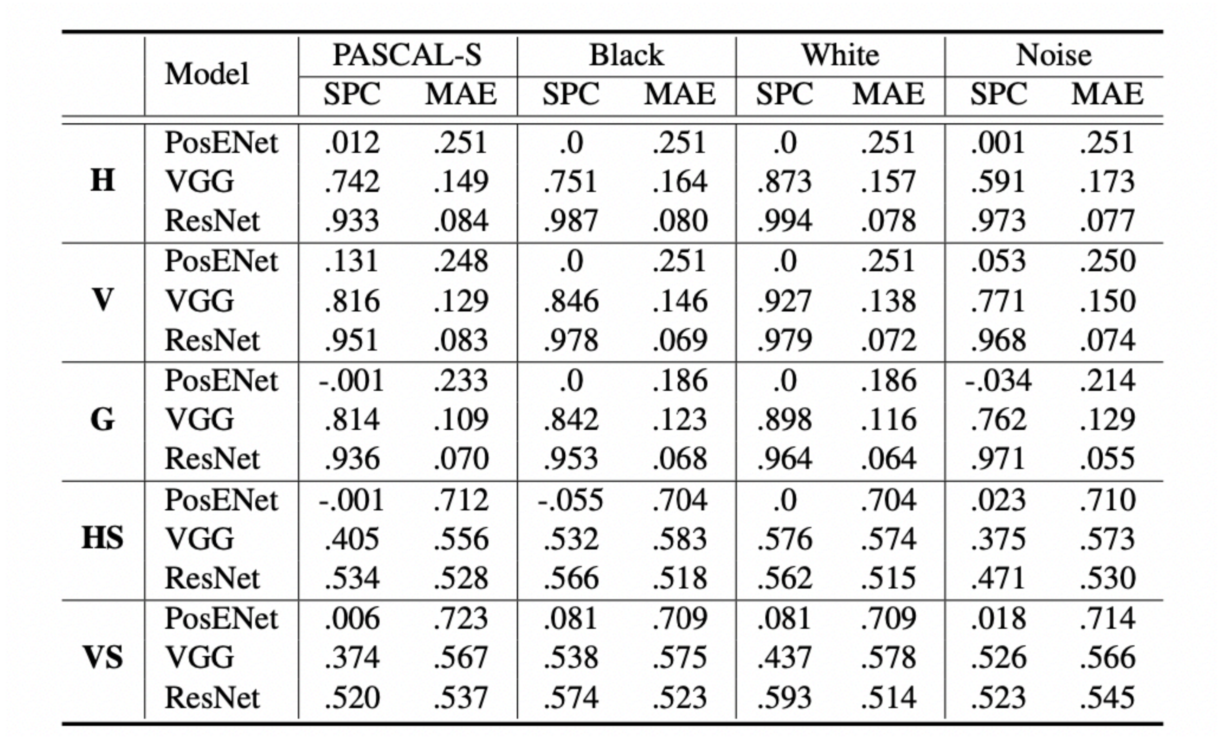
**Datasets used :** http://saliencydetection.net/duts/

Contains 10,533 images for training

**Evaluation metrics :** Spearmen correlation [-1,1] and Mean Absolute error

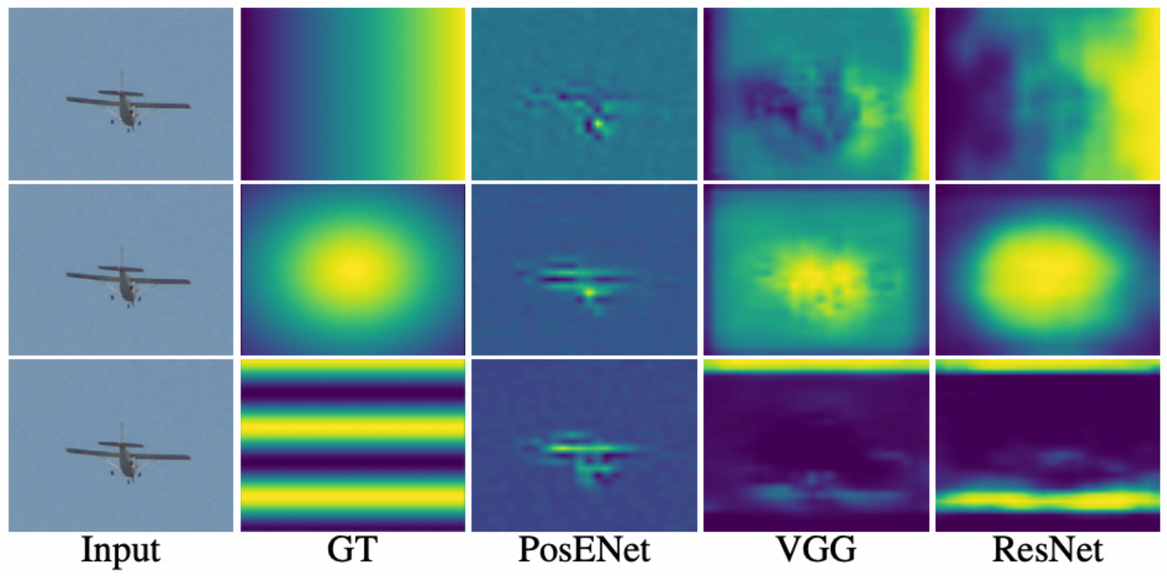
- In Position encoding branch, the layers are initialized with xavier intialization.
- Trained for 15 epochs, SGD optimizer.
- Five different types of Ground truths, hence 5 different models trained.
- Important : When training, the ground truth is always constant (either H, V, G, HS,VS)

**Results :**

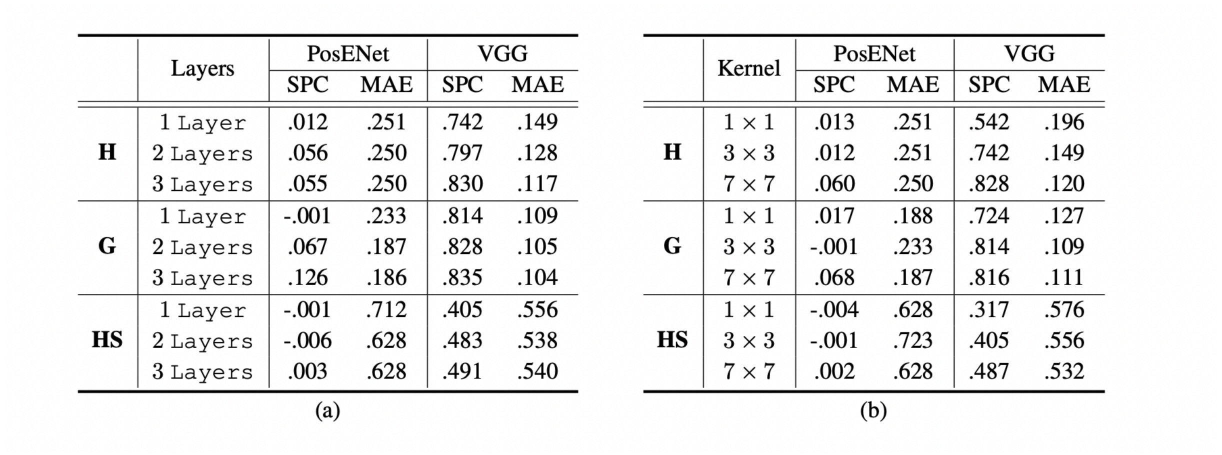


|  | Model | PASCAL-S |  | Black |  | White |  | Noise |  |
|---|---|---|---|---|---|---|---|---|---|
|  |  | SPC | MAE | SPC | MAE | SPC | MAE | SPC | MAE |
| **H** | PosENet | .012 | .251 | .0 | .251 | .0 | .251 | .001 | .251 |
|  | VGG | .742 | .149 | .751 | .164 | .873 | .157 | .591 | .173 |
|  | ResNet | .933 | .084 | .987 | .080 | .994 | .078 | .973 | .077 |
| **V** | PosENet | .131 | .248 | .0 | .251 | .0 | .251 | .053 | .250 |
|  | VGG | .816 | .129 | .846 | .146 | .927 | .138 | .771 | .150 |
|  | ResNet | .951 | .083 | .978 | .069 | .979 | .072 | .968 | .074 |
| **G** | PosENet | -.001 | .233 | .0 | .186 | .0 | .186 | -.034 | .214 |
|  | VGG | .814 | .109 | .842 | .123 | .898 | .116 | .762 | .129 |
|  | ResNet | .936 | .070 | .953 | .068 | .964 | .064 | .971 | .055 |
| **HS** | PosENet | -.001 | .712 | -.055 | .704 | .0 | .704 | .023 | .710 |
|  | VGG | .405 | .556 | .532 | .583 | .576 | .574 | .375 | .573 |
|  | ResNet | .534 | .528 | .566 | .518 | .562 | .515 | .471 | .530 |
| **VS** | PosENet | .006 | .723 | .081 | .709 | .081 | .709 | .018 | .714 |
|  | VGG | .374 | .567 | .538 | .575 | .437 | .578 | .526 | .566 |
|  | ResNet | .520 | .537 | .574 | .523 | .593 | .514 | .523 | .545 |

In the table above, posENet denotes just training with the PEM branch and not using feature maps from the encoder. We can clearly see that using just PEM has the worst performance. This means posENet can extract positional information consistent with ground truth position map only when it is paired with a deep learning encoder.



Impact of stacked CNN layers in the Position Encoding Module (initially used just 1 (3×3)) and impact of different kernel sizes (large kernel size → better resolve positional information).



| | Layers | PosENet | | VGG | |
|---|---|---|---|---|---|
| | | SPC | MAE | SPC | MAE |
| **H** | 1 Layer | .012 | .251 | .742 | .149 |
| | 2 Layers | .056 | .250 | .797 | .128 |
| | 3 Layers | .055 | .250 | .830 | .117 |
| **G** | 1 Layer | -.001 | .233 | .814 | .109 |
| | 2 Layers | .067 | .187 | .828 | .105 |
| | 3 Layers | .126 | .186 | .835 | .104 |
| **HS** | 1 Layer | -.001 | .712 | .405 | .556 |
| | 2 Layers | -.006 | .628 | .483 | .538 |
| | 3 Layers | .003 | .628 | .491 | .540 |

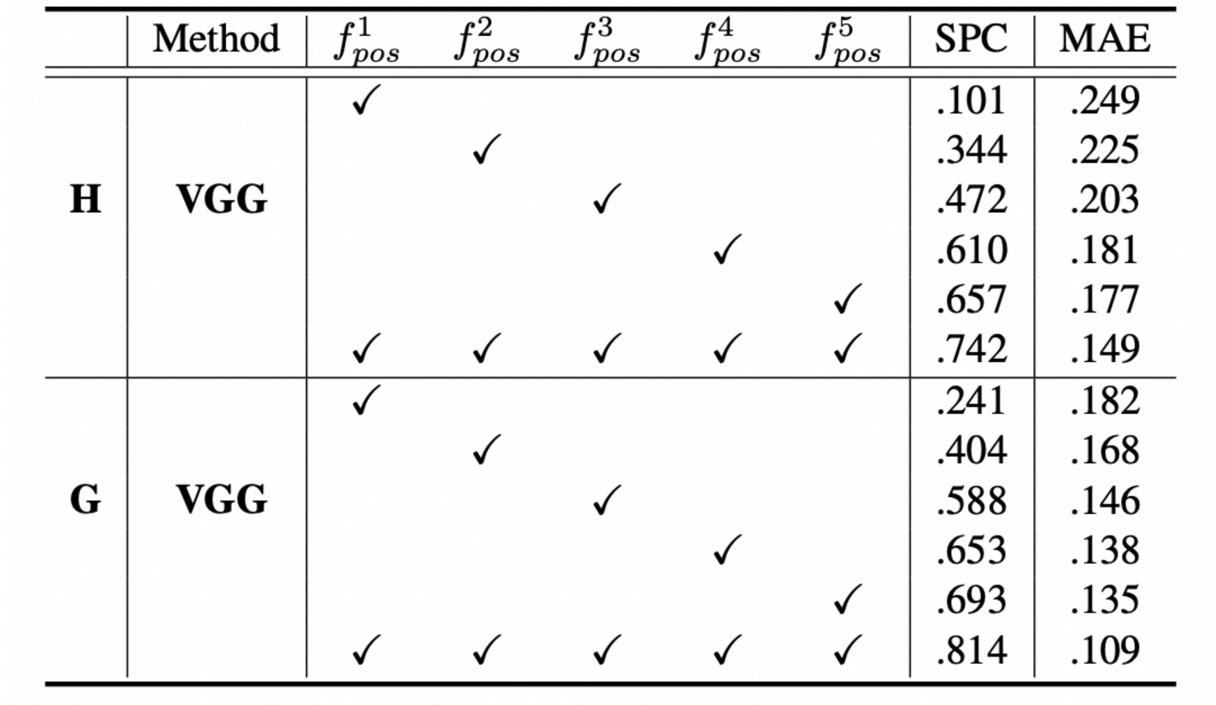
(a)

| | Kernel | PosENet | | VGG | |
|---|---|---|---|---|---|
| | | SPC | MAE | SPC | MAE |
| **H** | $1 \times 1$ | .013 | .251 | .542 | .196 |
| | $3 \times 3$ | .012 | .251 | .742 | .149 |
| | $7 \times 7$ | .060 | .250 | .828 | .120 |
| **G** | $1 \times 1$ | .017 | .188 | .724 | .127 |
| | $3 \times 3$ | -.001 | .233 | .814 | .109 |
| | $7 \times 7$ | .068 | .187 | .816 | .111 |
| **HS** | $1 \times 1$ | -.004 | .628 | .317 | .576 |
| | $3 \times 3$ | -.001 | .723 | .405 | .556 |
| | $7 \times 7$ | .002 | .628 | .487 | .532 |

(b)

Table 2: Quantitative comparison on the PASCAL-S dataset in terms of SPC and MAE with varying (a) number of layers and (b) kernel sizes. Note that (a) the kernel size is fixed to $3 \times 3$ but different numbers of layers are used in the PosENet. (b) Number of layers is fixed to one but we use different kernel sizes in the PosENet.

Next to examine where the positional information is actually stored in the encoder, the authors extracted features from the feature extraction blocks of the encoder and separately trained a position encoding network over it. The results are posted below.

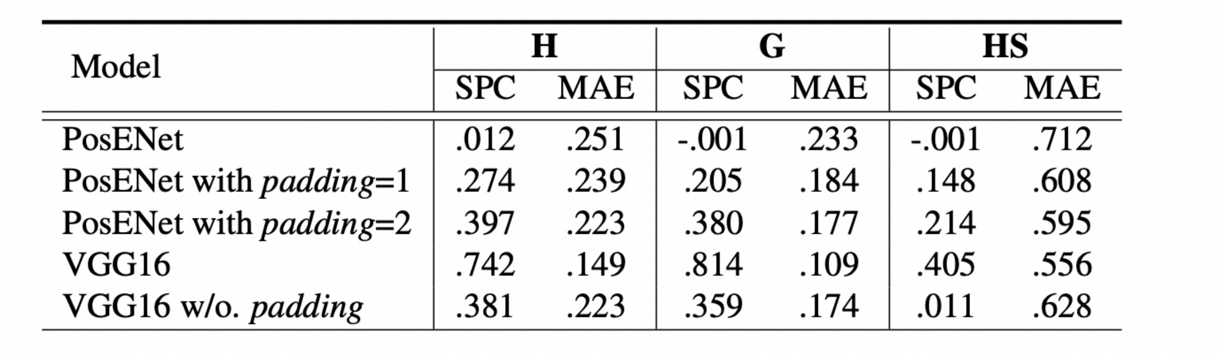


|  | Method | $f_{pos}^1$ | $f_{pos}^2$ | $f_{pos}^3$ | $f_{pos}^4$ | $f_{pos}^5$ | SPC | MAE |
|---|---|---|---|---|---|---|---|---|
| **H** | **VGG** | ✓ |  |  |  |  | .101 | .249 |
|  |  |  | ✓ |  |  |  | .344 | .225 |
|  |  |  |  | ✓ |  |  | .472 | .203 |
|  |  |  |  |  | ✓ |  | .610 | .181 |
|  |  |  |  |  |  | ✓ | .657 | .177 |
|  |  | ✓ | ✓ | ✓ | ✓ | ✓ | .742 | .149 |
| **G** | **VGG** | ✓ |  |  |  |  | .241 | .182 |
|  |  |  | ✓ |  |  |  | .404 | .168 |
|  |  |  |  | ✓ |  |  | .588 | .146 |
|  |  |  |  |  | ✓ |  | .653 | .138 |
|  |  |  |  |  |  | ✓ | .693 | .135 |
|  |  | ✓ | ✓ | ✓ | ✓ | ✓ | .814 | .109 |

We can see that, the features extracted from the deeper feature extraction block performs the best, which can be because, the number of feature maps extracted from the deeper layers is higher and it also indicates that, stronger encoding of positional information is present in the deeper layers of the network.

## Where does the Positional information come from?

The padding (Zero padding)applied at the borders deliver the positional information to learn to the encoder. To validate this hypothesis, a **vgg** net with all padding removed  and  initialized with the imagenet weights was used as encoder for the posENet. It was observed from the results that, **vgg** without padding performed poorly than a **vgg** with padding. Furthermore, the position information was introduced to the Position Encoding Module, by adding a zero padding (no padding in the beginning) and performance of the model was observed.



| Model | H | | G | | HS | |
|---|---|---|---|---|---|---|
| | SPC | MAE | SPC | MAE | SPC | MAE |
| PosENet | .012 | .251 | -.001 | .233 | -.001 | .712 |
| PosENet with *padding*=1 | .274 | .239 | .205 | .184 | .148 | .608 |
| PosENet with *padding*=2 | .397 | .223 | .380 | .177 | .214 | .595 |
| VGG16 | .742 | .149 | .814 | .109 | .405 | .556 |
| VGG16 w/o. *padding* | .381 | .223 | .359 | .174 | .011 | .628 |