# Augmenting Positional Encodings to CNNs for Saliency Detection

Danqi Liao
Princeton University
dl33@princeton.edu

## Abstract

*It's commonly believed that filters in Convolutional Neural Networks(CNNs) only learn what they are looking at but not where they are looking at. However, recent research [6] shows that absolute positional information is implicitly encoded and exploited in CNNs. It's shown that limiting CNNs' ability to exploit absolute location information significantly diminishes their performance in location dependent tasks such as saliency detection, semantic detection and objection detection. Intuitively, it's helpful for CNNs to learn both the semantic features (what) and absolute positions (where) for various location-sensitive image tasks. A natural question arises: Can we explicitly encode absolute positional information in CNNs to help boost the performance? What are the effects of explicitly encoding positional information in CNNs?In this work, we investigate the effects of augmenting positional information to CNNs, specifically in the task of saliency detection. We perform a set of comprehensive experiments with various ways to encode and augment positional information in Fully Convolutional Networks[13], and evaluate models in a wide range of datasets. Our experiments offer insights into how the utility of augmenting positional encodings changes with the original model's ability to do saliency detection. We also find that augmenting positional information encodes strong priors in the models.*

## 1. Introduction

Filters in Convolutional Neural Networks(CNNs) are commonly believed to only contain semantic information but not global position information. Recent research[6][7] shows that not only absolute positional information is encoded in CNNs, but also is exploited by CNNs to make predictions. [6] shows that deeper stacked CNNs with more layers are more capable for recovering positional information. [6] and [7] hypothesize that zero-padding and borders of images play a significant role in propagating absolute spatial information over the whole image, and [6] shows CNNs with padding significantly outperform these without padding in semantic segmentation and saliency detection.

Motivated by these recent findings that suggest CNNs learn and rely on positional information to a greater extent than commonly expected, we investigate the effects of explicitly augmenting positional information to CNNs, specifically in the context of saliency detection. We choose the task of saliency detection because networks need to attend to both the semantic content and spatial cues of the images to achieve good results in saliency detection[14]. To avoid using fully connected layers. we choose Fully Convolutional Network[13] as the network architecture for all experiments.

We experimented with five ways to encode absolute positional information, including Gaussian, horizontal position, vertical position, concatenation and summation of both horizontal and vertical positions. We also experimented with two ways to augment positional encodings to Fully Convolutional Networks: augmenting in encoder side and augmenting in decoder side. Since skip-connection layer is widely used to detect features in multi-scale and multi-levels[5], we also experimented with models with and without skip-connection layer.

We empirically show that directly encoding positional information in FCNs helps performance in a very limited extent. Some types of positional encodings such as Gaussian and HV tend to give a small amount of performance boost on saliency detection compared to baselines. We find that the utility of augmenting positional information to FCNs is dependent on the original network's ability to do saliency tasks; positional encodings are more effective when applied to weaker models such as FCNs without skip-connection than to relatively stronger models such as FCNs with skip-connection. Moreover, we find that augmenting positional information encodes strong priors in the models.

## 2. Related Work

### 2.1. Position Augmentation in CNNs

Although recent research[6][7] has shown that absolute position information is implicitly encoded in CNNs, few attempts have been made in the areas of explicitly encoding

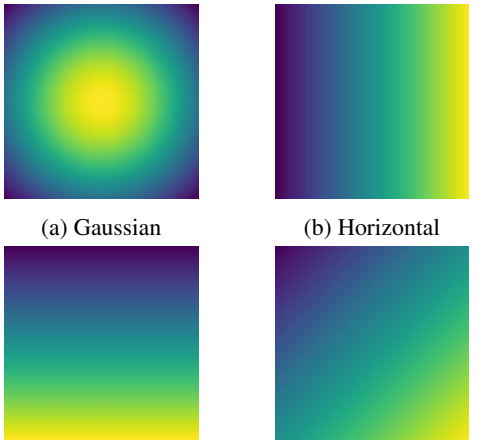
positional information in CNNs.

To the best of our knowledge, [18] and [11] are the first few works in attempting to directly encode positional information in CNNs. [18] experiments with three ways to encode positional information and evaluates them on the tasks of saliency detection, objection detection and scene parsing. Our work is partly similar to it but we differentiate from it by focusing on and going more in-depth with saliency detection. Moreover, [18] only experiments FCNs without skip-connection layer and only uses three datasets whereas our experiments evaluate models with and without skip-connection on five public available datasets which aligns more with the common practices in saliency detection literature[14]. In addition, [18] doesn't have a baseline that augments random information to the models so it's hard to pinpoint from which the performance gains actually stem, since the gain may come from models benefiting from more parameters. [18] only benchmarked and reported F-measures whereas we report both F-measures and Mean Absolute Error(MAE) and show both quantitative and qualitative results. Moreover, while [18] concludes that adding location information as extra channels significantly improves model's accuracy, our experiments show that the improvement is very limited and the utility of augmenting positional information changes with the original model's ability on the tasks.

[11] finds an interesting failing of CNNs on its inability to solve trivial coordinates transform problem and proposes to augment both vertical and horizontal coordinates to input RGB images to address this issue. Their experiments on toy MNIST dataset show that augmenting input images with coordinates can improve significantly on the specific object detection task. Their experiments on MNIST dataset is interesting but further experimentation on augmenting location in natural images instead of just simple synthetic MNIST images is needed to see if augmenting location is still effective and practical on different and more realistic datasets.

### 2.2. Saliency Detection

Saliency detection aims to map the most visually distinctive object(s) in an image. To achieve good results in saliency detection, models need to understand both the global semantic content of the whole image as well as the detailed spatial structures of the objects. Traditional saliency detection system relies on hand-crafted features to detect salient objects in input images[2]. To improve detection accuracy, many Fully Convolutional Networks[13] based networks are proposed [2] to leverage both the semantic information from deep stacked layers and detailed structure information from shallow fine layers.



(a) Gaussian (b) Horizontal

(c) Vertical (d) Horizontal+Vertical

Figure 1: Proposed four positional encodings

## 3. Augmenting Positional Encodings

To investigate the effects of explicitly encoding positional information in CNNs, we propose five ways to encode global positional information of pixels in images and two ways to add such positional encodings in Fully Convolutional Networks.

### 3.1. Encode Positional Information

- **Gaussian encodings**: We propose Gaussian encodings to explicitly encode the distance between pixels and center of the images. We use the following formula to construct a 2D Gaussian map with the center of the image as the origin.

$$E(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp -\frac{x^2 + y^2}{2\sigma^2} \tag{1}$$

  The intuition for Gaussian encodings is that salient objects tend to be around the center of images and adding information on how far each pixel is away from the center of image can help the model to predict whether pixels belong to part of the objects or the background.

  In our implementation, the sigma is chosen to be 110 so that pixels near the boundary can have non-zero values in $224 \times 224$ images. We also normalize the whole encoding matrix so that all pixel values sum up to 1.0.

- **Horizontal encoding**: We propose Horizontal encoding to directly encode the horizontal position of each pixels in input images. We use the following formula to construct 1D horizontal row with leftmost pixel as the origin. Assume W is the input image width, x is pixel's horizontal coordinate:

$$E(x) = \frac{x}{\sum_{x=0}^{x=W-1} x} \tag{2}$$

and we assign the same 1D horizontal encodings for each row to construct 224 × 224 2D positional encodings.

- **Vertical encodings**: We propose Vertical encodings to directly encode the vertical position of each pixels in input images. We use the following formula to construct 1D vertical column with top pixel as the origin. Assume H is the input image height, y is pixel's vertical coordinate:

$$E(y) = \frac{y}{\sum_{y=0}^{y=H-1} y} \tag{3}$$

and we assign the same 1D vertical encodings for each column to construct 224 × 224 2D positional encodings.

- **Concatenation of horizontal & vertical encoding**: To combine both the horizontal and vertical position information, we propose to concatenate both encodings to construct a 2 channel 2D positional encoding.

- **Summation of horizontal & vertical encodings**: Another way to combine both the horizontal and vertical positional information is to sum up the two encodings to construct a 1 channel 2D positional encodings. This summation encoding is diagonally symmetric as pixels symmetric to the dialog of the image have the same encoding values.

### 3.2. Augment Positional Information

We experiment with two ways to add positional information to Encoder-Decoder based Fully Convolutional Networks[1]. One way is to augment positional encoding in encoder side as extra channel(s) to input RGB images. The other way is to augment positional encoding in decoder side as extra feature(s) to the input of last convolutional layer. The two ways of augmenting is illustrated below.

### 3.3. Network Architecture

We choose VGG11[16] as the encoder architecture and a sequence of transposed convolutional layers as the decoder. We experiment two variations of this Encoder-Deconder architecture: one without skip connection and one with skip connection. The architecture and activation size are summarized in the following table:

### 3.4. Loss Function

We use standard binary cross entropy loss in training the models. Assume $P(i, j)$ is the predicted probability of pixel at (i,j) being salient object pixel, then the loss $\ell$ for size $H \times W$ input image with target salient map $y$ is computed as the following:

$$\ell = \frac{\sum_{i,j} y_{i,j} \log P(i,j) + (1 - y_{i,j}) \log(1 - P(i,j))}{H \times W}$$

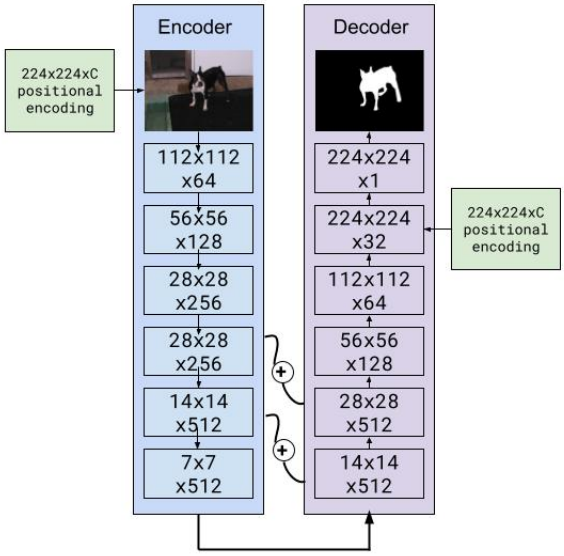



Figure 2: Encoder-Decoder based FCN architecture with two ways to augment positional encoding: (left side) augment on the encoder side as extra channel(s) in input images; (right side) augment on the decoder side as extra feature(s) to the input of last convolutional layer.

| Layer | Activation Size |
|---|---|
| input | 3 × 224 × 224 |
| 64×3×3 Conv, ReLU | 64 × 224 ×224 |
| 2×2 MaxPool with stride 2 | 64 × 112 ×112 |
| 128×3×3 Conv, ReLU | 128 × 112 × 112 |
| 2×2 MaxPool with stride 2 | 128 × 56 ×56 |
| 256×3×3 Conv, ReLU | 256 × 56 × 56 |
| 256×3×3 Conv, ReLU | 256 × 56 × 56 |
| 2×2 MaxPool with stride 2 | 256 × 28 ×28 |
| 512×3×3 Conv, ReLU | 512 × 28 × 28 |
| 512×3×3 Conv, ReLU | 512 × 28 × 28 |
| 2×2 MaxPool with stride 2 | 512 × 14 ×14 |
| 512×3×3 Conv, ReLU | 512 × 14 × 14 |
| 512×3×3 Conv, ReLU | 512 × 14 × 14 |
| 2×2 MaxPool with stride 2 | 512 × 7 ×7 |

Table 1: Encoder Architecture: VGG11.

| Layer | Activation Size |
|---|---|
| input | 512 × 7 × 7 |
| 512×3×3 Transpose Conv, BatchNorm | 512 × 14 × 14 |
| 256×3×3 Transpose Conv, BatchNorm | 256 × 28 × 28 |
| 128×3×3 Transpose Conv, BatchNorm | 128 × 56 × 56 |
| 64×3×3 Transpose Conv, BatchNorm | 64 × 112 × 112 |
| 32 ×3×3 Transpose Conv, BatchNorm | 32 × 224 × 224 |
| 1 ×1×1 Conv | 1 × 224 × 224 |

Table 2: Decoder Architecture: all transpose convolutional layer has stride 2 and output padding 1.

# 4. Experimentation

## 4.1. Datasets

We train all models on MRSA-B dataset[12] and evaluate trained models on five most frequently used datasets in saliency detection tasks, including DUTS-TE[17], ECSSD[20], HKU-IS[9], PASCAL-S[10] and DUT-OMRON[15]. MRSA-B dataset[12] contains 5000 images with diverse categories and context. Most images in MRSA-B only involve single salient object. We specifically choose MRSA-B to benchmark with the experiments done by [18] since they train all their models on MRSA-B. DUTS-TE is the test data of DUTS dataset [17]. DUTS-TE is collected from the ImageNet DET test set[3] and the SUN data set[19] and contains complex saliency detection images. ECSSD[20] is another challenging dataset that has many semantically meaningful and structurally complex evaluation images. HKU-IS[9] contains 4447 challenging images, most of which have either multiple salient objects or low contrast and overlapping between image boundaries. PASCAL-S[10] is collected from PASCAL VOC 2010 segmentation validation dataset[4]. Although PASCAL-S only has 850 test images but it contains very difficult and complex scenarios for saliency detection with most of images have complex salient objects and scenes. DUT-OMRON[15] is a large size dataset that include very challenging evaluation images.

## 4.2. Implementation & Experiments Setup

We train our models on the MSRA-B dataset and evaluate them on the rest five datasets listed above. Before training, the input images are resized to 224 × 224 and normalized with mean 0.485, 0.456, 0.406 and standard deviation 0.229, 0.224, 0.225. We use the Adam optimizer[8] to train our network with learning rate $1e^{-}4$ and no warmup steps nor weight decaying. We train our models with 30 epochs and batch size 22. We choose 30 epochs because the experiments show that train loss converges within 30 epochs. The sigma for Gaussian encodings is 110. It takes around 40 minutes to train a single model and evaluate trained models on the five datasets mentioned above. We train and evaluate each model variant five times to compute the average performance and report the model with the performance that's closest to the corresponding average score. We evaluate each model variant five times and take the average because the individual experiment results between different runs of the same model have a variance in MAE ranging from 0.5 to 1.3. We hypothesize the variance comes from the dense structure of the prediction output. In total, it takes more than 100 hours to train and evaluate all models. During inference, we resize the input image and do the image normalization similar to the training image preprocessing procedure and feed the normalized image to the model to produce saliency map.

We implement our models and experiments using Pytorch framework and train all models on AWS EC2 virtual machines with a single Tesla V100 GPU. Our FCN encoder is adapted from https://github.com/pochih/FCN-pytorch.

## 4.3. Evaluation Metrics

We choose three metrics for evaluating our models across all datasets: Precision-Recall (PR) curve, F-measure and MAE (Mean Absolute Error).

we plot the Precision-Recall (PR) curve. The precision and recall are computed by comparing the binarized predicted map and the ground truth. The PR Curve is plotted by comparing the precision and recall under different thresholds.

F-measure is a combined measure for precision and recall. Specifically here we compute $F_\beta$ using following formula:

$$F_\beta = \frac{(1+\beta^2) \times \textit{precision} \times \textit{recall}}{\beta^2 \times \textit{precision} + \textit{recall}}$$

We choose $\beta^2$ to be 0.3 here, following the practice of most saliency detection literature on addressing the importance of precision of the detection systems.

MAE is the average absolute difference per pixel between predicted saliency map and the ground truth. Given the output of models $P(i,j)$ the predicted saliency map, and ground truth map $G(i,j)$, the MAE is computed as following:

$$MAE = \frac{1}{W \times H} \sum_{i}^{W} \sum_{j}^{H} \mid P(i,j) - G(i,j) \mid$$

where $W \times H$ is the size of the input image and (i,j) denotes the coordinate of pixels.

## 4.4. Results

**Positional encodings at encoder without skip-connection.** We present our evaluation results in Table 3.

| | DUTS-TE | | HKU-IS | | ECSSD | | PASCAL-S | | DUT-OMRON | |
|---|---|---|---|---|---|---|---|---|---|---|
| model | $F_\beta$ | *MAE* | $F_\beta$ | *MAE* | $F_\beta$ | *MAE* | $F_\beta$ | *MAE* | $F_\beta$ | *MAE* |
| baseline | 0.6189 | 0.1351 | 0.7477 | 0.1296 | 0.7554 | 0.1500 | 0.7050 | 0.1792 | 0.6376 | 0.1310 |
| random#1 | 0.6061 | 0.1444 | 0.7345 | 0.1405 | 0.7411 | 0.1623 | 0.6851 | 0.1878 | 0.6292 | 0.1402 |
| random#2 | 0.6077 | 0.1451 | 0.7375 | 0.1422 | 0.7373 | 0.1657 | 0.6853 | 0.1896 | 0.6346 | 0.1405 |
| Gaussian | 0.5989 | 0.1269 | 0.7465 | 0.1134 | 0.7612 | 0.1336 | 0.7030 | 0.1719 | **0.6212** | 0.1212 |
| Horizontal | 0.6060 | 0.1319 | 0.7473 | 0.1224 | 0.7627 | 0.1415 | 0.7040 | 0.1756 | 0.6372 | 0.1246 |
| Vertical | 0.6108 | 0.1234 | 0.7538 | 0.1134 | 0.7596 | 0.1363 | 0.6970 | 0.1742 | 0.6354 | 0.1178 |
| HV | **0.6202** | **0.1194** | **0.7605** | **0.1082** | **0.7683** | **0.1299** | **0.7127** | **0.1693** | 0.6414 | **0.1140** |
| H+V | 0.6085 | 0.1300 | 0.7467 | 0.1227 | 0.7541 | 0.1449 | 0.6960 | 0.1781 | 0.6404 | 0.1233 |

Table 3: Evaluation results for augmenting positional encodings in encoder without skip-connection. Best scores are highlighted in bold. Random#1 is the model augmented with random extra channel. Random#2 is the random model with 2 extra channels to benchmark against position augmented model with 2 extra channels. HV denotes the models with concatenation of both horizontal and vertical encodings. H+V is the model with the summation of horizontal and vertical Encoding.

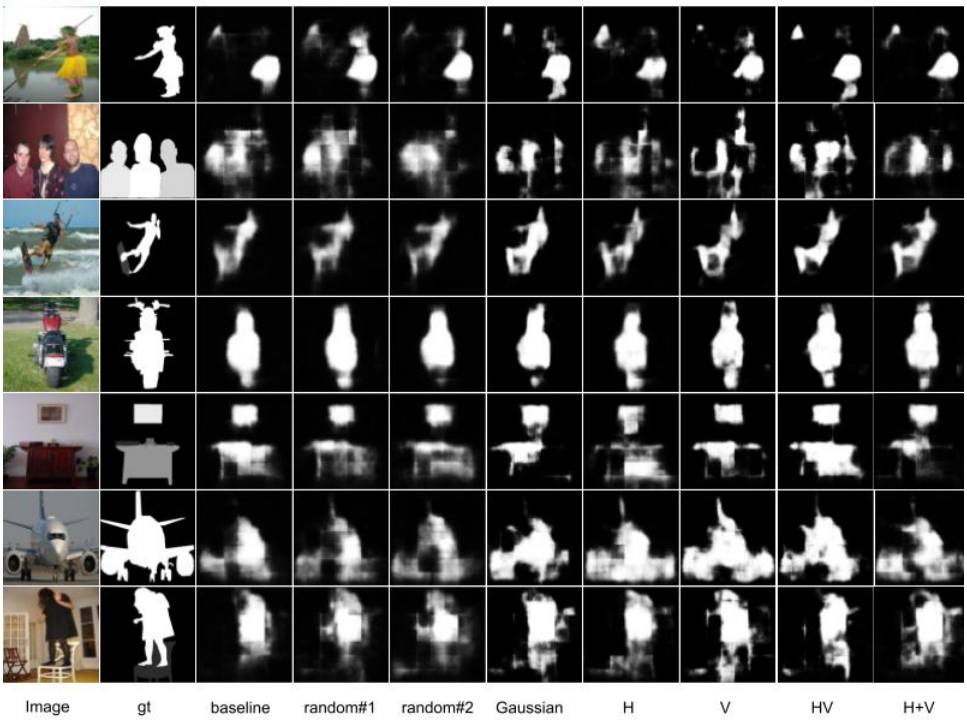



Figure 3: Visual comparisons of different models.

Model HV, augmenting input image with both horizontal and vertical encodings, outperforms all other models across all five datasets except DUT-OMRON, improving 1.7% compared baseline, 2.7% compared to random#1 in average MAE. The close second is model Gaussian augmented with Gaussian encodings. The third best is Vertical, fourth best is H+V, the summation of both horizontal and vertical encodings. All baseline and position augmented models outperform both random#1 and random#2.

All position augmented models outperform the random baseline and the standard baseline. This result suggests that augmenting positional information in input image is helping the model's performance on saliency detection to some extent in this experimentation setup, and simply augmenting input image with extra channels doesn't help model performance.

From the PR curve (Figure 4), we observe some interesting behaviors of position augmented models compared to the baseline and randomly augmented models: the precision of position augmented models is lower-bounded by a

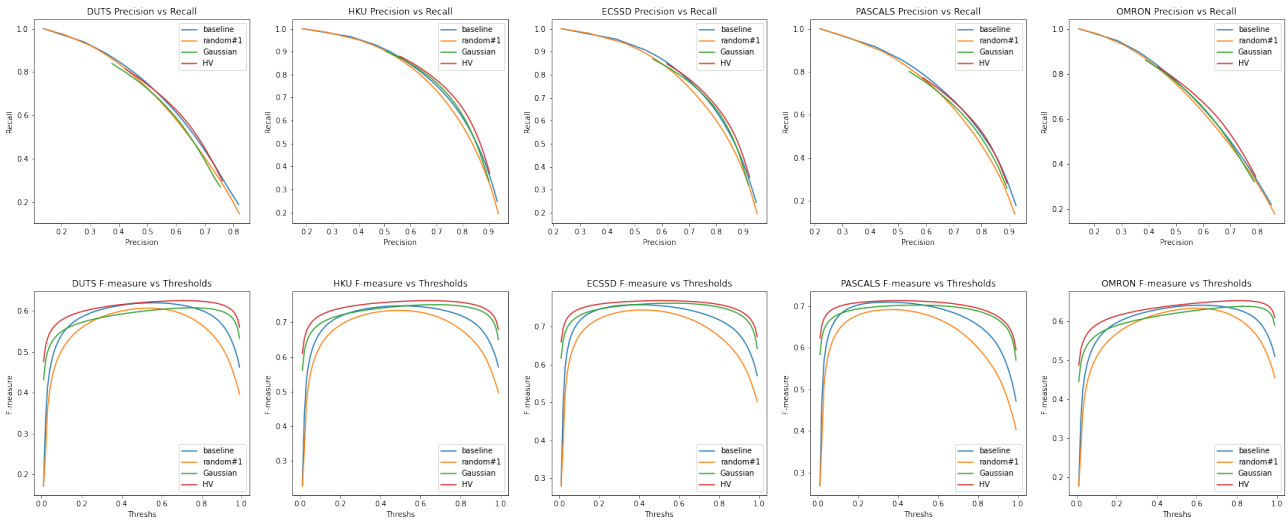



Figure 4: PR Curve and F-measure vs Thresholds for baseline, random#1, Gaussian and HV models

larger number, ranging from 0.4 to 0.6, compared to randomly augmented models and the baseline. This larger lower bound of precision suggests that the position augmented models are very sure some pixels are not salient pixels no matter how small the threshold is. On the other hand, the precision of position augmented model is also upper-bounded by a smaller number than that of baseline and randomly augmented models. The smaller upper bound of precision suggests that the position augmented models are very sure some negative pixels are positive pixels no matter how large the threshold is. The position augmented models are above the baseline and random model in HKU-IS, ECSSD and PASCAL-S F-measure plots and the F-measure of model HV is above that of all other models across all datasets (Figure 4). The F-measure of position augmented models is lower-bounded by a larger number than that of baseline and random models. The trend displayed by the PR curve and F-measure curve suggests that there are some stronger priors built into position augmented models than the baseline and random models. In this specific experimentation, the extra priors seem to give position augmented models an extra edge over the random and baseline models.

We also present a visual comparison of different models on some images from the evaluation datasets (Figure 3). Comparing the visualization of baseline and model Gaussian, it seems like the saliency map generated from model Gaussian is more confident near center of object region and is more condident about the general outline and shape of the objects. For example, for example image at row 2 with 3 people in the image (Figure 3), the saliency map generated by the baseline spreads out the positive pixels across a wide region whereas the one generated by model Gaussian concentrates the positive pixels around the center of each individual person. The map generated by model H places more positive pixels on the horizontal direction and the one generated by model V places more positive pixels on the vertical direction.

**Positional encodings at encoder with skip-connection.** We present our evaluation results in Table 4. Model Gaussian and baseline are the best models with Gaussian having a very small edge on dataset HKU-IS and ECSSD over baseline. Model random#2 performs better than model H+V, V and random #1.

Skip-connection gives a performance boost to all models including the baseline and position augmented models (Table 3 vs Table 4). The performance difference between Gaussian and baseline in average is almost trivial in skip-connection settings, suggesting that the baseline with skip-connection may already have implicitly learned a good representation of image's global positional information, and having access to lower level features in images helps the model learn global positional information of the image.

Baseline and Gaussian models have very similar PR curve (Figure 5), almost completely overlapping with each other across five different datasets. Interestingly, the precision of HV model has a slightly larger lower-bound than that of Gaussian and baseline, and the precision of random model has the largest range among that of other models. It may suggest that slightly stronger priors are encoded in model HV than in Gaussian, baseline and random model. F-measure of Gaussian and baseline model has the largest lower-bound among all models and is less sensitive to varying thresholds.

| | DUTS-TE | | HKU-IS | | ECSSD | | PASCAL-S | | DUT-OMRON | |
|---|---|---|---|---|---|---|---|---|---|---|
| model | $F_\beta$ | *MAE* | $F_\beta$ | *MAE* | $F_\beta$ | *MAE* | $F_\beta$ | *MAE* | $F_\beta$ | *MAE* |
| baseline | 0.6373 | 0.1142 | 0.7831 | 0.1020 | 0.7845 | 0.1253 | **0.7221** | **0.1657** | 0.6587 | 0.1093 |
| random#1 | 0.6236 | 0.1190 | 0.7653 | 0.1125 | 0.7658 | 0.13701 | 0.6918 | 0.1768 | 0.6567 | 0.1126 |
| random#2 | 0.6341 | 0.1157 | 0.7686 | 0.1095 | 0.7770 | 0.1336 | 0.6990 | 0.1736 | 0.6601 | 0.1102 |
| Gaussian | 0.6368 | 0.1140 | **0.7856** | **0.1005** | **0.7870** | **0.1235** | 0.7169 | 0.1674 | **0.6557** | **0.1096** |
| Horizontal | **0.6436** | 0.1127 | 0.7820 | 0.1053 | 0.7776 | 0.1310 | 0.7060 | 0.1718 | 0.6722 | 0.1067 |
| Vertical | 0.6376 | 0.1193 | 0.7766 | 0.1104 | 0.7755 | 0.1343 | 0.7114 | 0.1720 | 0.6668 | 0.1128 |
| HV | 0.6357 | **0.1137** | 0.7741 | 0.1069 | 0.7634 | 0.1350 | 0.6992 | 0.1733 | 0.6598 | 0.1088 |
| H+V | 0.6260 | 0.1170 | 0.7745 | 0.1073 | 0.7681 | 0.1337 | 0.7025 | 0.1728 | 0.6480 | 0.1126 |

Table 4: Model evaluation results for augmenting position in encoder with skip-connection layers

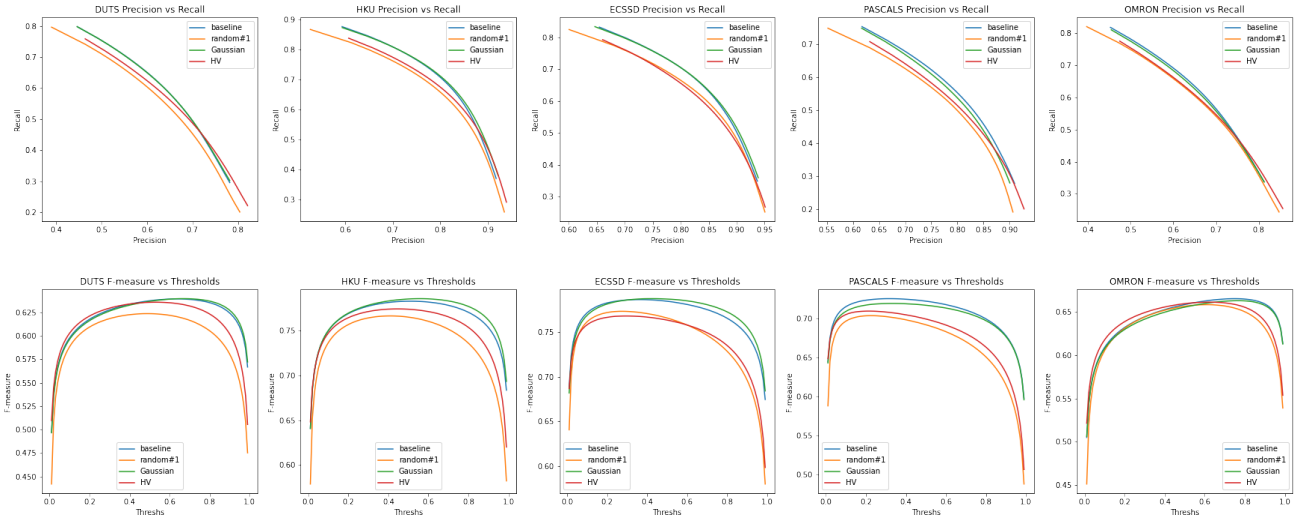



Figure 5: PR Curve and F-measure vs Thresholds for augmenting position in encoder with skip-connection layers

**Positional encodings at decoder.** We present our evaluation results in Table 5. Model HV is the best model and the close second is model Gaussian. Model HV improves 1.5% in average MAE, compared to the baseline. The improvement is only half of the improvement in the encoder setting (1.5% vs 2.7%). Interestingly, except HV and Gaussian model, all other position augmented models perform worse than the baseline, although all position models still perform better than the randomly augmented models.

The precision of both model HV and Gaussian has a larger lower bound than baseline and random models (Figure 6). The F-measure of both model HV and Gaussian is above that of other models and has a much tighter range compared to baseline and random models.

Decoder augmented models seem less effective than encoder augmented in our experiments. The performance discrepancy may come from the fact that encoder augmented models have more extra weights than decoder augmented models ($64\times3\times3$ vs $1\times3\times3$). Since position information in decoder augmented models only go through layers of convolution and batch normalization, the diminished improvement may also suggest that positional signals can be utilized more effectively when applied on non-linear transformation (in the case of encoder augmented models) than linear transformation.

**Positional encodings at decoder with skip-connection.** The result is very similar to that of encoder with skip-connection experiments. Gaussian and baseline are the best models with model HV being the close third. Model H+V and V perform worse thanr model random#1 and random#2.

## 5. Conclusion

In this work we investigate the effects of augmenting positional information to Fully Convolutional Network for saliency detection tasks. We provide an in-depth experimentation for various ways to encode and augment posi-

| | DUTS-TE | | HKU-IS | | ECSSD | | PASCAL-S | | DUT-OMRON | |
|---|---|---|---|---|---|---|---|---|---|---|
| model | $F_\beta$ | *MAE* | $F_\beta$ | *MAE* | $F_\beta$ | *MAE* | $F_\beta$ | *MAE* | $F_\beta$ | *MAE* |
| baseline | **0.6189** | 0.1351 | 0.7477 | 0.1296 | 0.7554 | 0.1500 | 0.7050 | 0.1792 | 0.6376 | 0.1310 |
| random#1 | 0.6052 | 0.1517 | 0.7364 | 0.1457 | 0.7461 | 0.1654 | 0.6959 | 0.1880 | 0.6344 | 0.1460 |
| random#2 | 0.6044 | 0.1377 | 0.7274 | 0.1383 | 0.7302 | 0.1616 | 0.6696 | 0.1890 | 0.6384 | 0.1319 |
| Gaussian | 0.6152 | **0.1199** | 0.7558 | **0.1098** | 0.7669 | 0.1309 | 0.7101 | 0.1693 | 0.6384 | 0.**1142** |
| Horizontal | 0.6137 | 0.1390 | 0.7446 | 0.1354 | 0.7337 | 0.1620 | 0.6874 | 0.1855 | **0.6418** | 0.1340 |
| Vertical | 0.6147 | 0.1443 | 0.7365 | 0.1431 | 0.7326 | 0.1669 | 0.6851 | 0.1893 | 0.6350 | 0.1403 |
| HV | 0.6119 | 0.1237 | **0.7578** | 0.1099 | **0.7700** | **0.1301** | **0.7161** | **0.1677** | 0.6369 | 0.1171 |
| H+V | 0.6046 | 0.1447 | 0.7344 | 0.1422 | 0.7288 | 0.1677 | 0.6833 | 0.1894 | 0.6193 | 0.1422 |

Table 5: Model evaluation results for augmenting position in decoder without skip-connection layers

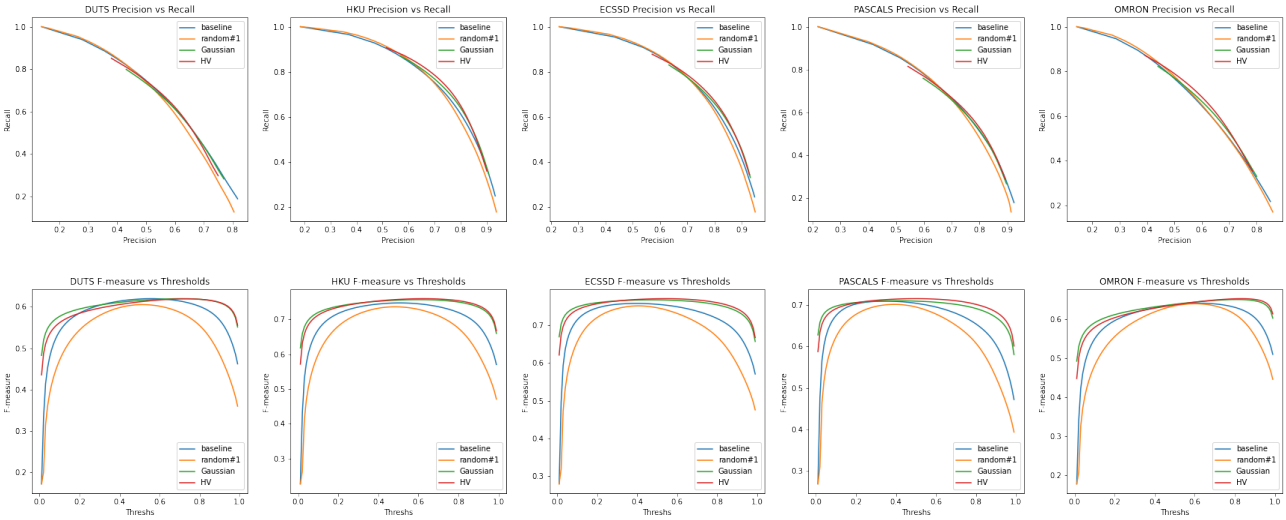



Figure 6: PR Curve and F-measure vs Thresholds for augmenting position in decoder without skip-connection layers

tional information in networks across a variety of datasets. Our study shows that directly encoding position information in FCNs helps performance in a limited extent. Positional encodings is more effective when applied to weaker models such as FCN without skip-connection. Directly encoding positional information in FCNs gives the network a stronger priors but the utility of that augmented prior varies depending on different encoding methods and model architectures. Since we only experiment with augmenting location information in CNNs in the context of saliency detection, future work and experimentation are needed to fully investigate the effects of directly encoding positional information in CNNs in the settings of other image tasks such as semantic segmentation, objection detection and scene parsing.

## References


[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017. 3

[2] Ali Borji, Ming-Ming Cheng, Huaizu Jiang, and Jia Li. Salient object detection: A benchmark. *IEEE Transactions on Image Processing*, 24(12):5706–5722, Dec 2015. 2

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 4

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html. 4

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016. 1

[6] Md Amirul Islam, Sen Jia, and Neil DB Bruce. How much position information do convolutional neural networks encode? *arXiv preprint arXiv:2001.08248*, 2020. 1

[7] Osman Semih Kayhan and Jan C van Gemert. On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14274–14285, 2020. 1

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. 4

[9] Guanbin Li and Yizhou Yu. Visual saliency based on multiscale deep features. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015. 4

[10] Yin Li, Xiaodi Hou, Christof Koch, James M. Rehg, and Alan L. Yuille. The secrets of salient object segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun 2014. 4

[11] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. *arXiv preprint arXiv:1807.03247*, 2018. 2

[12] Tie Liu, Jian Sun, Nan-Ning Zheng, Xiaoou Tang, and Heung-Yeung Shum. Learning to detect a salient object. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007. 4

[13] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1, 2

[14] Xuebin Qin, Zichen Zhang, Chenyang Huang, Chao Gao, Masood Dehghan, and Martin Jagersand. Basnet: Boundary-aware salient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2

[15] Xiang Ruan, Na Tong, and Huchuan Lu. How far we away from a perfect visual saliency detection-dut-omron: a new benchmark dataset. In *Korea-Japan Joint Workshop on Frontiers of Computer Vision, Okinawa, Japan*, 2014. 4

[16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. 3

[17] Lijun Wang, Huchuan Lu, Yifan Wang, Mengyang Feng, Dong Wang, Baocai Yin, and Xiang Ruan. Learning to detect salient objects with image-level supervision. In *CVPR*, 2017. 4

[18] Zhenyi Wang and Olga Veksler. Location augmentation for cnn. *arXiv preprint arXiv:1807.07044*, 2018. 2, 4

[19] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492, 2010. 4

[20] Qiong Yan, Li Xu, Jianping Shi, and Jiaya Jia. Hierarchical saliency detection. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1162, 2013. 4