Overlap Training to Mitigate Inconsistencies Caused by Image Tiling in CNNs*

Yu An[1]✉, Qing Ye[2], Jiulin Guo[3], and Ruihai Dong[1]

[1] Insight Centre for Data Analytics, University College Dublin, Dublin, Ireland
yu.an@insight-centre.org
[2] Key Laboratory of Tectonics and Petroleum Resources, Ministry of Education, China University of Geosciences, Wuhan, China
[3] C&C Reservoirs, Brunel House, Reading, United Kingdom

Abstract. This paper focuses on the problem of inconsistent predictions of modern convolutional neural networks (CNN) at patch (i.e. sub-image) boundaries. Limited by the graphics processing unit (GPU) resources, image tiling and stitching countermeasure have been applied for most megapixel images, that is, cutting images into overlapping tiles as CNN input, and then stitching CNN outputs together. However, we found that stitched (i.e. recovered) predictions have discontinuous grid-like noise. We propose a simple yet efficient overlap training framework to mitigate the inconsistent prediction at patch boundaries without changing the model architecture while improving the stability, robustness of the model. We have applied our solution to various CNNs (such as U-Net, DeepLab, RCF) and tested them on two real-world datasets. Extensive experiments suggest that the new framework is sufficient in reducing inconsistency and outperform these countermeasures. The source code and coloured figures are made publicly available online at: https://github.com/anyuzoey/Overlap-Training.git

Keywords: Convolutional Neural Networks · Computer Vision · Image Segmentation · Fault Recognition

1 Introduction

Convolutional neural networks (CNNs) and its recent variations have led to extraordinary performances in various computer vision tasks. In general, CNN models take image pixel as inputs and use multiple convolutional layers and pooling layers to capture image features. The intermediate layers need to be stored in the CNN calculation process. The memory size required by the model is proportional to the number of input pixels [29]. Limited by the memory capacity of the graphics processing unit (GPU), in many scenarios, CNN cannot directly use the entire original size image as input, especially megapixel images (e.g. seismic images). There are two main countermeasures: one of them is to down-sample the input image, but it will considerably lose local details. Not suitable for

* Supported by Science Foundation Ireland SFI/12/RC/2289_P2

tasks such as fault recognition that make extensive use of local details. Another prevailing countermeasure is to cut out the original size image as sub-images (usually called patches) as model inputs and stitch the outputs.

We found that this image cutting off-dealing-stitching approach will create discontinuous and grid-like noise (see black arrows in Fig. 1) on the stitched outputs of several trained fault recognition networks. Because the patch predictions on the top, bottom, or left and right are inconsistent, the original continuous geological fault is marked as discontinuous. Geological faults are the planar or gently curved fractures in the earth's crust. The relative displacement of rocks on opposite sides of the fracture is usually related to the occurrence of earthquakes [11]. Fault recognition is a critical process for seismic data interpreters to understand the subsurface geological structure. For seismic data interpreters, fault recognition (i.e. identify and label certain types of discontinuous reflections) is a manual process, which highly depends on data quality and experience, and usually takes several months.

The discontinuities and grid-like noise greatly affect the user experience of the fault recognition application. A single threshold cannot fix it as it can not effectively distinguish between noise and potential faults among all data. Since the grid-like noise only appears at the boundary position, we call it the boundary effect. These effects are underestimated and rarely discussed in the literature. Only a few relevant studies have suggested methods (proved using their datasets) that might attenuate the boundary effect: use the largest possible patch size [14, 22]; pad extra zeros [15, 17]; or increase the number of overlapping pixels [14, 25]. These methods are either very complicated to use and cannot be widely applied, or add a considerable redundancy and cannot fundamentally solve the problem. We will review and discuss these methods in detail in section 2.

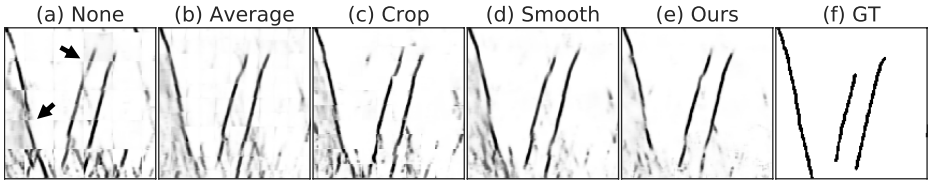

Fig. 1. Examples of different image stitching techniques on the Thebe dataset. (a) None, images are cropped into non-overlapping patches as CNN input. The final prediction after stitching can clearly distinguish the traces of image stitching. (b) Average, the prediction of pixels in overlapping areas is averaged. (c) Crop, crop the boundary pixels of the patch according to the number of the overlapping pixels. (d) Smooth, the prediction of each patch first is multiplied by 2D spine window (i.e. position-related 2D weight) (e) Ours (f) Manual interpolation by an expert as ground truth. Coloured vision is available in the code link.

In this paper, we propose a novel overlap training framework to mitigate the boundary effect that we discovered in fault recognition CNNs. The framework

uses a simple yet efficient overlap constraint to handle the boundary information better, to improve consistency of CNN prediction in the boundary position, and also the overall performance of the fault recognition application. Our framework obtained the state-of-the-art performance compared with related solutions (see Section 4.4, 4.5).

The rest of this article is organised as follows: Section 2 reviews related literature on tiling and stitching techniques, potential causes of boundary effects, and the recommended solutions. Section 3 details our proposed framework, the datasets used and the CNN architectures we tested. In Section 4, we provide experimental results and analysis of the proposed method and then summarise in Section 5.

2 Related Work

Since the breakthrough of the AlexNet [19] in the 2012 ILSVRC (ImageNet Large-Scale Visual Recognition Challenge), the CNN has achieved the state-of-the-art results in various computer vision tasks (such as image classification [12, 31], image segmentation [5–8], edge detection [20, 32]). With the development of imaging technology and explosive growth of data, we have encountered more and more large-scale image processing tasks (e.g. remote sensing image processing). Nevertheless, it is unlikely to store large-scale intermediate layers because of the limited GPU resources. For higher accuracy, it is commonly recommended to crop overlapping tiles (i.e. patches) as inputs, and then use averaging [2, 24, 25] or weighted sum [9, 27] or cropping to stitch model predictions. Fig. 1 provides an example of different image stitching methods on the fault recognition dataset. Details of the dataset will be introduced later in section 3.1.

Improvements made by applying these image stitching approaches imply that model outputs are not consistent in the overlapping area or more specifically, the boundary area. This situation conflicts with the common assumption that CNN is translation equalvariance or sometimes called translation invariance (that is, CNN can successfully identify the target object, regardless of the position of the object in the image.) [10, 16, 18]. Moreover, this also means that simple post-processing like image stitching can not fundamentally find and solve the problem.

A few recent studies link this boundary effect with zero padding [14, 15, 17] and non-unary stride (i.e. stride > 1) operation [3, 14] in CNNs. Zero-padding is the default setting for convolutional layers, which pad zeros on the boundary of the input feature map to control output dimensions. If no zero is padded (i.e. 'valid' padding), convolution output will be smaller than the input dimension. 'Same' padding is used to make convolution output remains the same dimensions as input. In convolution layers, boundary pixels are less supported than the relative centre pixels [17]. For instance, a 3×3 'same' padding convolution, one output pixel is the dot product between the learnable kernel and corresponding nine pixels in input feature maps. Thus, for the top-left pixel, the corresponding nine pixels contain five zeros. Therefore, when more convolution stacked, zeros

are propagated closer and closer to centre pixels, which result in larger boundary effect [17]. For non-unary stride operation, such as 2×2 Max-pooling operation, that is, downsampling the input by selecting the maximum value among four adjacent pixels, will directly lose information and even change output value for shifted inputs.

Some researches recommended using the largest possible patches to reduce boundary effect, because the larger the patch size, the fewer zero-padding pixels compared to the total patch pixels [14, 22]. However, the improvement in model accuracy may not be mainly caused by the reduction of zero-padding, but is related to a larger overlapping area. Besides, there are also tasks like our fault recognition application, in which very large patch size may result in a less generalised model.

From the perspective that CNN can exploit absolute spatial location, two state-of-the-art studies discuss similar boundary effects and also point to zero-padding [15, 17]. [15] proposed that CNN obtains absolute spatial information through zero-padding to improve accuracy in position-related tasks. Their experiments proved that increase padding value in convolution layer from 0 to 2 could consistent improve model accuracy. Similar but different, [17] proposed that absolute spatial information can be removed by add zero padding. They named this solution as 'FULL' convolution. For the standard 3×3 convolution, 'FULL' convolution is equivalent to padding=2. Authors suggested replacing 'same' padding or 'valid' padding to 'FULL' padding and add extra zero-padding for residual connections to avoid encoding absolute spacial location. This solution is proved to give higher accuracy on image classification tasks and image matching tasks. Although the main conclusions of the two studies conflict, they both agree that extra zero-padding can improve model performance. In this paper, we compared our proposed framework with this zero-padding solution in Section 4.4 and achieved better performance. Besides, our framework is a more flexible solution that does not require modification of the model architecture, including the padding settings in convolution layers and the input size.

3 Proposed Framework

In this section, we describe our proposed overlap training framework in detail, see Fig. 2. In general, pixel-level classification problems (e.g. image segmentation) use batches of paired images and masks to train CNNs. The difference between the predictions and the masks are used to update parameters through back-propagation. However, the model trained in this standard approach does not predict consistently at the overlap region of two adjacent patches. We believe that although the absolute position of the target object in the two overlapping patches shifted, the model output should follow the facts, that is, the overlapping region should obtain consistent predictions. Therefore, we propose a constraint to limit the difference between the two predictions in the overlapping region. We construct pairs of overlapping images as inputs and add an overlap loss as the constraint to force the model to predict the overlapping region more

consistently. More precisely, in the image preprocessing stage, we prepare pairs of left and right overlapping input images ($Input_{left}$ and $Input_{right}$) and their corresponding masks (GT_{left} and GT_{right}), and the overlapping part accounts for half of the patch size. Thus, $4 \times batchsize$ images are generated at each iteration. A batch of $Input_{left}$ and a batch of $Input_{right}$ are sequentially inputted into one CNN model to obtain predictions $Pred_{left}$ and $Pred_{right}$, respectively. In addition to calculating loss ($loss_{left}$ and $loss_{right}$) between the model prediction and the corresponding masks, we also define and add an $loss_{overlap}$ that will compare the differences between the model predictions in the overlapping area (i.e. $Overlap_{left}$ and $Overlap_{right}$). Here, we introduced two hyper-parameters: N and α, where N means that the overlap loss is added only after the N^{th} epoch, and α represents the weight of the overlap loss. The total loss l_{total} is the weighted sum of three losses, see Eq. 1.

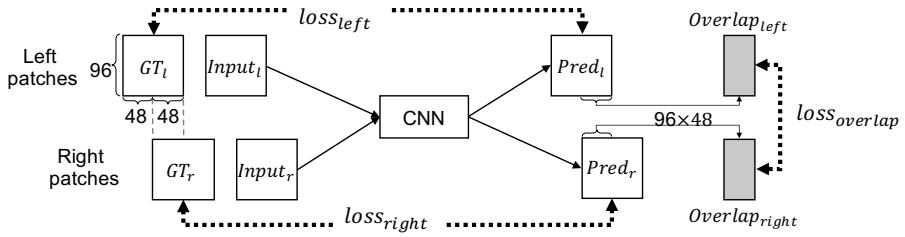

Fig. 2. The Overlap Training Framework

The overlap loss we added may have a certain degree of conflict with the standard loss ($loss_{left}$ and $loss_{right}$). We observe that if the overlap loss is not added, the difference of the predictions in the overlap region between two adjacent patches will increase and then level off. The reason for this phenomenon is that in the early stage of training, the model has not yet learned useful features (weights are close to zeros), resulting in the model output close to zero, just like random noise. Therefore, prematurely adding the overlap loss is equivalent to continuous attack model using a large amount of noise or wrong labels. Thus, the convergence of the model is delayed. However, we also do not recommend adding the overlap loss too late. Firstly, it will make the training time longer. Secondly, one of the purposes of adding overlap loss is to alleviate the overfitting of the model, therefore to improve its generalisation. If the overlap loss is added too late, the model cannot be prevented from overfitting because it is already overfitting. Therefore, we recommend using a hyper-parameter to indicate the appropriate time step to add the overlap loss.

$$loss_{total} = \begin{cases} loss_{left} + loss_{right} + \alpha loss_{overlap}, & epoch > N \\ loss_{left} + loss_{right}, & otherwise \end{cases} \tag{1}$$

3.1 Datasets

The fault recognition dataset: Thebe is a large-scale 3-dimensional public dataset, which contains the original seismic reflection dataset of the Thebe gas field in the NW shelf of Australia and the corresponding fault labels labelled by experts from Fault Analysis Group, University College Dublin. This dataset contains 1,803 consecutive grey-scale seismic images and corresponding mask image (i.e. fault labels) with height and width of 1,537 and 3,174 pixels. Each seismic image sampled at 12.5-meter intervals is a vertical slice of the subsurface. Faults are labelled by polylines and converted to masks with a line width of approximately 8 pixels. First 900 pairs of seismic images and masks are the training set, the following 200 pairs are the validation set, and the last 703 pairs are the test set.

In addition to the primary fault recognition dataset, we also apply our proposed framework on a medical dataset, purely for testing its generalisation ability instead of pursuing a state-of-the-art result. It is a publicly available blood vessel segmentation dataset called DRIVE (Digital Retinal Images for Vessel Extraction) [30]. It is selected because a blood vessel is thin (similar to faults) and relatively sensitive to boundary effect than other segmentation objects. This dataset contains 40 coloured (RGB) retinal images with 768×584 pixels (a reasonable size for a medical dataset). Each image has a circular field of view (FOV) with a diameter of approximately 540 pixels. Only pixels within the FOV are considered when evaluation. This group of 40 images has been divided into training and test sets, each containing 20 images. Test set labels are not publicly available, and test set performance can be acquired by upload prediction to the official website. In our experiment, we split the 20 training images into the train (16 images) and validate set (4 images) for hyper-parameter search.

3.2 Applied Convolutional Neural networks

For fault recognition, we consider it as either image segmentation or edge detection task. Thus, we apply this framework to these two types of CNNs (image segmentation models: U-Net [23] and DeepLab V3+ [5–8] with a MobileNet V2 [13, 26] encoder, edge detection model: RCF (Richer convolution features) [20]) In this paper, we will observe whether our proposed framework is valid on these three models and whether they have the same improvement trend. We briefly reviewed the three models mentioned above and introduced our modifications.

The U-Net model is a classic image segmentation network, which removes the fully connected layer of the classic VGG16 network [28] as an encoder, and then adds a symmetric decoder and shortcuts to locate the boundary positions accurately. Here, we remove a few convolution layers and reduce feature maps in each convolution layer because our input patch size is much smaller than the original U-Net input size. A sigmoid layer is added at the end of the network to give soft classification outputs range from 0 to 1.

The DeepLab V3+ model is a advanced image segmentation network [8]. It stacks more convolutional layers and uses the atrous spatial pyramid pooling (ASPP) to obtain long-distance information, which implies a larger reception

field, thereby improving the accuracy of the model. A larger reception field, however, may lead to a more significant boundary effect. Limited by GPU resources, we select MobileNet V2 as the backbone encoder. Similarly, a sigmoid layer is added to the outputs of this model to give target object probability.

The RCF model is a classic edge detection network, which is also modified from the classic VGG16 network and another edge detection network called HED (Holistic edge detection) [32]. The classic VGG16 network contains five stages of convolution layers divided by four Max-pooling layers and some fully-connected layers. RCF replaces these fully-connected layers by up-sample layers, convolution layers and one sigmoid layer. The model has a total of six outputs, including five prediction maps up-sampled to the input dimension from the feature map obtained from the convolutional layers of each stage, and another prediction map obtained by fusing the five prediction maps. All six feature maps learn parameters by weighted binary cross-entropy. The last output is used as the final prediction output of the model.

3.3 Overlap Loss

Binary cross-entropy (Eq. 2) is used for all three CNN models listed above. l_n stands for loss of the n^{th} pixel, w_n represents the weights. $\hat{y}_n$ stands for model prediction while y_n stands for the ground truth value (0 or 1). Here, w_n is equal to 1 for all pixels (n) in the two segmentation models, while RCF use Eq. 3 to calculate weights. In Eq. 3, Y stands for the total number of pixels of the input image. $|Y_-|$ and $|Y_+|$ represents negative pixels (i.e. label=0) and positive pixels (i.e. label=1) respectively.

$$l_n = -w_n \left[y_n \cdot \log \hat{y}_n + (1 - y_n) \cdot \log \left(1 - \hat{y}_n\right) \right] \tag{2}$$

$$w_n = \begin{cases} |Y_+|/|Y|, \, y_n = 0 \\ |Y_-|/|Y|, \, y_n = 1 \end{cases} \tag{3}$$

In our overlap training framework, $loss_{left}$ and $loss_{right}$ in Fig. 2 remain the original binary cross-entropy. Since they are log loss, we define the $loss_{overlap}$ as a log version of mean absolute error (see Eq. 4) so that the range for hyperparameter search should be small. Also, we believe a symmetric loss will be beneficial as the boundary position in one patch is the centre position (i.e. better predicted) in the opposite patch. It can be easily implemented using the detach function in PyTorch, see Eq. 5.

$$logMAE = -log(1 - abs(prediction - target)) \tag{4}$$

$$\begin{aligned} loss_{overlap} = {} & logMAE(Overlap_{left}, Overlap_{right}.detach()) \\ & + logMAE(Overlap_{right}, Overlap_{left}.detach()) \end{aligned} \tag{5}$$

4 Experiments

In this section, we first detail the general training settings of the experiments. Then, the evaluation method used is introduced. Finally, three different experiments were designed to systematically and comprehensively test and analyse the performance of our overlap training framework.

4.1 Training Setup

For a fair comparison, model setting like input size (96×96 pixels), overlap size (48 pixels), batch size (128 individual images if without overlap while 64 pairs if with overlap), validation loss monitors: learning rate (lr) scheduler (factor=0.1, patient=5) and early stopping scheduler (patient=10) is consistent among all experiments. Optimiser settings (Adam for segmentation network with initial lr of 0.01, SGD optimiser for RCF with initial lr of 1e-6 and momentum of 0.9, weight decay of 0.0002) is consistent for the same architecture. Unless specified, random seed is 1. All images are recovered by smooth stitching [9] and evaluated on the area of interest. We selected the hyper-parameters N and α that obtained the best validation set results through a grid search. The last layer is used to calculate the overlap loss for the edge detection model.

Thebe dataset is a highly imbalanced dataset. The expert only label faults that they are certain to be geological faults in the area of interest. A sliding window of size 96×96 with a stride of 48 pixels and a filter (fault pixels $> 3\%$) are used to generate the left patches. The corresponding right patches are obtained by right shifting left patches 48 pixels. Overall, $181,029 \times 2$ and $64,317 \times 2$ pairs (i.e. image and mask) of patches as the training set and the validation set, respectively.

For the DRIVE dataset, we divide the original training set (20 images) into the train (16) and validation (4) set. We augment each instance three times using basic methods: horizontal flip, vertical flip, rotate 180°. We also augment each training instance using the commonly used Contrast-Limited Adaptive Histogram Equalization (CLAHE) transform [4, 21]. A sliding window of 96×96 pixels with a stride of 24 pixels is used to generate more patches. Overall, $33,440 \times 2$ half overlap pairs of patches and $6,688 \times 2$ pairs of patches are generated for training and validating sets. Since DRIVE dataset is an RGB dataset, we modify U-Net and DeepLab V3+ to take 3 channel input instead of 1 channel. We did not modify model architecture or generate more training patches to optimise the test set performance.

4.2 Evaluation Metrics

We apply the evaluation method used by the public edge detection BSDS500 dataset [1], which can give comprehensive summarise of soft classification (0 to 1 probability) performance. Given 99 thresholds, this evaluation method calculates precision, recall, F1 score 99 times for each test set image. Thereby, eight evaluation results can be obtained, which are Threshold, Recall, Precision, fixed

contour threshold (ODS), Best_Recall, Best_Precision, per-image best threshold (OIS), average precision (AP). The first four metrics are optimal on the entire test set level while the next three represents the average of the per-image optimal precision, recall, F1 score, respectively. AP is the area under the precision-recall curve (AUCPR). Three global evaluation results ODS, OIS, AP, are the primary evaluation metrics for comparison. The ground-truth of the DRIVE test set, however, is not disclosed, test set evaluation result (i.e. mean, maximum, minimum F1 score) is obtained by uploading binary masks.

4.3 Exp1: Can Overlap Training Improve Accuracy, Stability?

The primary focus of this framework is to improve the performance of our fault recognition application. Here, we hypothesis that using overlap training framework can improve model accuracy and stability. Model stability is evaluated by the standard deviation of the three random initialisation. The benchmarks for this experiment (named with the suffix of '_w/o') are the models that have not been trained with our framework. The experimental group, that is, the models obtained by overlap training, are named suffix of '_w'

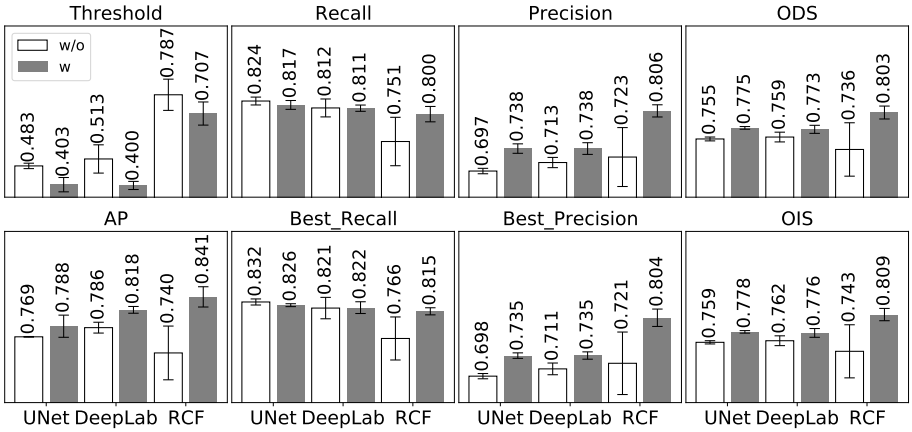

Fig. 3. Exp1: Thebe test set performance

According to Figure 3, overlap training can bring consistent accuracy improvements among the three architectures. For both image segmentation architectures (U-Net and DeepLab), the use of overlap training framework has improved all global metrics, including ODS by about 2.7% and 1.8%, OIS by 2.5% and 1.9%, and AP by 2.5% and 4.1%. The overlap training framework is particularly prominent for the improvement of the model RCF. The two recall metrics and the two precision metrics are significantly improved by approximately 6.5% and 11.5%, respectively. These improvements lead to a significant improvement

of the global evaluation metrics ODS and OIS by approximately 9%, and metric AP rise from 0.74 to 0.841, soaring by almost 14%. Besides, our framework also improves model stability, which is proved by the consistent decreases of standard deviations. According to visual examples in Fig. 4, our framework can provide cleaner predictions even when compared to smooth stitching.

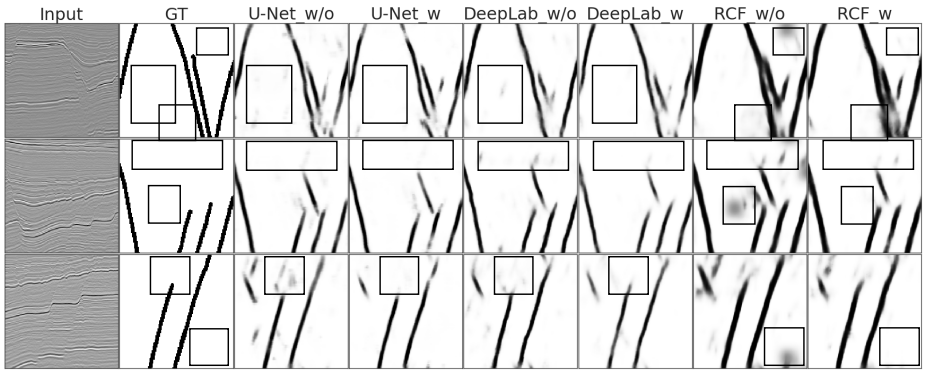

Fig. 4. Exp1: Three Thebe test set examples of the Thebe dataset.

To test the generalisation ability of our proposed framework, we also perform the same experiment on the DRIVE dataset. Since the DRIVE dataset is an image semantic segmentation problem, we only tested two segmentation architectures, namely U-Net and DeepLab. As shown in Table 1, for model U-Net, overlap training increases the average F1 value by 4%, in which the maximum F1 value is increased by 2%, and the minimum F1 value is increased by more than 7%. Overlap training does not improve DeepLab as significantly as U-Net on the DRIVE dataset. The average, maximum, and minimum F1 values are improved by 1%, 0.2%, and 0.8%, respectively. This experiment once again validates our conclusion that using overlap training framework can train models with higher accuracy and better robustness.

Table 1. Exp1: DRIVE test set performance

Model	**F1 Mean**	F1 Max	F1 Min
U-Net_w/o	0.7650	0.8318	0.6389
U-Net_w	**0.7984**	**0.8492**	**0.6858**
DeepLab_w/o	0.7829	0.8313	0.7628
DeepLab_w	**0.7905**	**0.8329**	**0.7688**

4.4 Exp2: How Overlap Training Affect Boundary Inconsistency?

We designed a mean absolute error versus distance experiment to investigate how the overlap training framework improves the boundary effect and how it performs compared to the state-of-the-art solution. We define the distance is the maximum value of the horizontal and vertical distance from the patch centre, see Fig. 5. Here, we use all non-overlap patches from the area of interest of the Thebe test set. For each architecture, the baseline (i.e. no overlap training) is drawn with a dashed line, and the model trained with overlap framework is shown with a solid line, see Fig. 6. The dot lines, which named with "fullconv", are the state-of-the-art solution suggested by the related work [15, 17]. The "fullconv" solution set zero-padding=2 for every standard 3×3 convolution and pad extra zeros before all residual connections. Due to the large number of residual connections used in the DeepLab MobileNet v2 [26], change to "fullconv" solution is very complicated, so we have not modified this architecture.

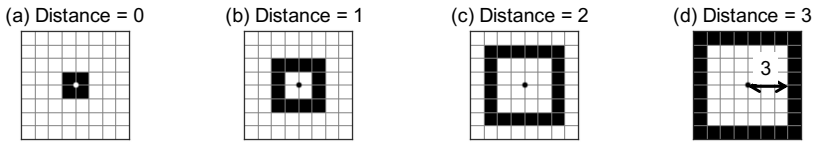

Fig. 5. Exp2: An 8×8 pixels patch example of distance masks. The distance represents $Max(h_i, v_j)$, where h_i and v_j is the horizontal and vertical distance between pixel (i, j) to the patch centre (i.e. the dot). The pixels marked black are the pixels with the same corresponding distance from the centre of the patch. For each distance, the mean absolute error of all test set patches on the corresponding pixels is calculated.

According to Fig. 6, our overlap training method, obtained the lowest mean absolute error on all three architectures, especially at the patch boundary (i.e. around 30 pixels away from patch centre). Compared with the "FULL" convolution solution, our overlap training solution not only can easier apply to different models but also achieved lower mean absolute error.

4.5 Exp3: Is Overlap Training Better than Stitching?

We hypothesis that our overlap training framework improves consistently across different image stitching methods. We evaluate this hypothesis on both Thebe dataset and DRIVE dataset using architecture U-Net. According to Table 2 and 3, overlap training consistently improves test set performance among different image stitching methods, in which apply overlap training on crop stitching increase performance the most. In Table 2, overlap training significantly improves the minimum F1 score by 7.339%, 16.886%, 20.937% when applying smooth stitching, average stitching, crop stitching, respectively. Meanwhile, improve the maximum F1 score by 1-2% and improve the average F1 score by 2-6%. The

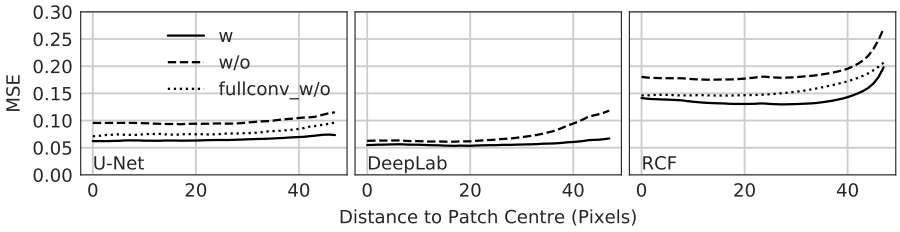

Fig. 6. Exp2: Mean absolute error versus the distance to the centre of the patch on the three models: U-Net (left), DeepLab(middle), RCF(right)

significant improvements on the minimum F1 score indicate our framework is more robust to outlier images.

Similarly, different image stitching method also improves Thebe test set performance, see Table 3. Although the image stitching method is different. The trends for all metrics except AP are the same. When applying overlap training, the test set optimal threshold decreased, recall, and best recall slightly decreased by less than 1%, precision and best precision increased 3-7%. Thus, two F1 scores, that are ODS and OIS increased 1-3%. Overall, these improvement trends indicate that our overlap training framework can train better and more robust models.

Table 2. Exp3: DRIVE test set performance across 3 image stitching method

Method	Model	**F1 Mean**	F1 Max	F1 Min
smooth	U-Net_w/o	0.7650	0.8318	0.6389
	U-Net_w	+4.366%	+2.095%	+7.339%
average	U-Net_w/o	0.7729	0.8385	0.6008
	U-Net_w	+2.424%	+1.110%	+16.886%
crop	U-Net_w/o	0.7496	0.8234	0.5805
	U-Net_w	+6.290%	+2.950%	+20.937%

5 Conclusion

In this paper, we focus on reducing the boundary effect discovered in the fault recognition application. The boundary effect caused by inconsistent predictions between adjacent overlapping tiles. A simple yet effective overlap training framework is proposed. This framework reduces the boundary effect by preparing pairs of left and right overlapping input patches and adding a constraint on the prediction difference of overlapping positions. Extensive experiments have proved

Table 3. Exp3: Thebe test set performance across 3 image stitching method

Method	Model	Threshold	Recall	Precision	**ODS**	Best_Recall	Best_Precision	**OIS**	**AP**
smooth	U-Net_w/o	0.4900	0.8288	0.6919	0.7542	0.8367	0.6929	0.7580	0.7691
	U-Net_4_0.5	-24.490%	-0.334%	+5.294%	+2.656%	-0.910%	+5.344%	+2.416%	-0.390%
average	U-Net_w/o	0.4500	0.8112	0.6955	0.7489	0.8221	0.6943	0.7528	0.7898
	U-Net_4_0.5	-24.444%	-0.250%	+3.585%	+1.779%	-0.348%	+3.339%	+1.618%	+0.164%
crop	U-Net_w/o	0.5100	0.8292	0.6639	0.7374	0.8400	0.6638	0.7416	0.7102
	U-Net_4_0.5	-23.529%	-1.012%	+7.650%	+3.618%	-0.937%	+7.014%	+3.353%	-1.866%

that our framework can provide consistent performance improvements on various CNNs (such as U-Net, DeepLab, RCF) and different domain datasets (such as Thebe, DRIVE). Finally, compared with state-of-the-art solutions that deal with the boundary effect, our overlap training framework is not only easier to apply to different models but also achieved smaller error.

References

1. Arbeláez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(5), 898–916 (5 2011). https://doi.org/10.1109/TPAMI.2010.161
2. Audebert, N., Saux, B.L., Lefèvre, S.: Semantic segmentation of earth observation data using multimodal and multi-scale deep networks (2016)
3. Azulay, A., Weiss, Y.: Why do deep convolutional networks generalize so poorly to small image transformations? Journal of Machine Learning Research **20**, 1–25 (2019)
4. Buslaev, A., Parinov, A., Khvedchenya, E., Iglovikov, V.I., Kalinin, A.A.: Albumentations: fast and flexible image augmentations (2018)
5. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected crfs (2014)
6. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs (2016)
7. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation (2017)
8. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation (2018)
9. Chevalier, G.: Smoothly-blend-image-patches (Aug 2017), https://github.com/Vooban/Smoothly-Blend-Image-Patches
10. Cohen, T.S., Welling, M.: Group equivariant convolutional networks (2016)
11. Fossen, H.: Structural Geology. Cambridge University Press (2010). https://doi.org/10.1017/CBO9780511777806
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015)
13. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications (2017)

14. Huang, B., Reichman, D., Collins, L.M., Bradbury, K., Malof, J.M.: Tiling and stitching segmentation output for remote sensing: Basic challenges and recommendations (2018)
15. Islam, M.A., Jia, S., Bruce, N.D.B.: How much position information do convolutional neural networks encode? (2020)
16. Kauderer-Abrams, E.: Quantifying translation-invariance in convolutional neural networks (2017)
17. Kayhan, O.S., van Gemert, J.C.: On translation invariance in cnns: Convolutional layers can exploit absolute spatial location (2020)
18. Kondor, R., Trivedi, S.: On the generalization of equivariance and convolution in neural networks to the action of compact groups (2018)
19. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. p. 1097–1105. NIPS'12, Curran Associates Inc., Red Hook, NY, USA (2012)
20. Liu, Y., Cheng, M.M., Hu, X., Wang, K., Bai, X.: Richer convolutional features for edge detection (2016)
21. Luo, Z., Zhang, Y., Zhou, L., Zhang, B., Luo, J., Wu, H.: Micro-vessel image segmentation based on the ad-unet model. IEEE Access **7**, 143402–143411 (2019)
22. Reina, G.A., Panchumarthy, R., Thakur, S.P., Bastidas, A., Bakas, S.: Systematic evaluation of image tiling adverse effects on deep learning semantic segmentation. Frontiers in Neuroscience **14**, 65 (2020). https://doi.org/10.3389/fnins.2020.00065, https://www.frontiersin.org/article/10.3389/fnins.2020.00065
23. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation (2015)
24. Roth, H.R., Oda, H., Zhou, X., Shimizu, N., Yang, Y., Hayashi, Y., Oda, M., Fujiwara, M., Misawa, K., Mori, K.: An application of cascaded 3d fully convolutional networks for medical image segmentation (2018)
25. Saito, S., Yamashita, T., Aoki, Y.: Multiple object extraction from aerial imagery with convolutional neural networks. Electronic Imaging **2016**(10), 1–9 (2016)
26. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks (2018)
27. Shi, Y., Wu, X., Fomel, S.: Saltseg: Automatic 3d salt segmentation using a deep convolutional neural network. Interpretation **7**, SE113–SE122 (04 2019). https://doi.org/10.1190/int-2018-0235.1
28. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014)
29. Siu, K., Stuart, D.M., Mahmoud, M., Moshovos, A.: Memory requirements for convolutional neural network hardware accelerators. In: 2018 IEEE International Symposium on Workload Characterization (IISWC). pp. 111–121. IEEE, Raleigh, NC (2018)
30. Staal, J., Abramoff, M., Niemeijer, M., Viergever, M., Ginneken, B.: Ridge-based vessel segmentation in color images of the retina. IEEE transactions on medical imaging **23**, 501–9 (04 2004). https://doi.org/10.1109/TMI.2004.825627
31. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks (2019)
32. Xie, S., Tu, Z.: Holistically-nested edge detection (2015)