Hotspot Detection via Attention-based Deep Layout Metric Learning

Hao Geng
CUHK

Haoyu Yang
CUHK

Lu Zhang
CUHK

Jin Miao
Synopsys

Fan Yang
Fudan Univ.

Xuan Zeng
Fudan Univ.

Bei Yu
CUHK

Abstract

With the aggressive and amazing scaling of the feature size of semiconductors, hotspot detection has become a crucial and challenging problem in the generation of optimized mask design for better printability. Machine learning techniques, especially deep learning, have attained notable success on hotspot detection tasks. However, most existing hotspot detectors suffer from suboptimal performance due to two-stage flow and less efficient representations of layout features. What is more, most works can only solve simple benchmarks with apparent hotspot patterns like ICCAD 2012 Contest benchmarks. In this paper, we firstly develop a new end-to-end hotspot detection flow where layout feature embedding and hotspot detection are jointly performed. An attention mechanism-based deep convolutional neural network is exploited as the backbone to learn embeddings for layout features and classify the hotspots simultaneously. Experimental results demonstrate that our framework achieves accuracy improvement over prior arts with fewer false alarms and faster inference speed on much more challenging benchmarks.

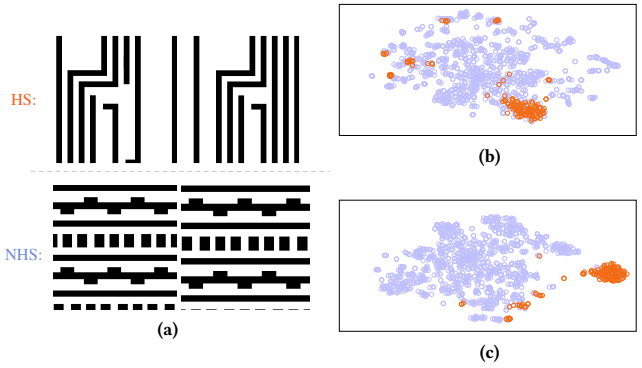

Figure 1: The snapshots of layout clips from ICCAD12 benchmarks [15] and Barnes-Hut t-SNE [16] visualizations of feature embeddings on the same benchmarks: (a) The examples of hotspots and non-hotspots; (b) The DCT feature embeddings of TCAD'19 [10]; (c) The feature embeddings of our proposed framework. "HS" is used to represent hotspot clips, while "NHS" refers to non-hotspot clips.

1 Introduction

As the technology node of integrated circuits scales down to 7*nm* and beyond, the techniques for lithographic processes are supposed to manage the ever-shrinking feature size. However, owing to the delayed progress of lithography techniques, lithographic processes variations emerge during the manufacturing, and thus lead to yield loss. For example, lower fidelity patterns on a wafer (a.k.a. lithography hotspot) is one of the crucial issues. Many techniques of early detection for hotspots have been proposed to ensure manufacturability. The prevalently applied lithography simulation technique could attain high accuracy, nevertheless, it is also known to be pretty time-consuming. Therefore, two other types of quick detection methods are proposed as alternatives to lithography simulation. One is based on pattern matching [1–3], and the other is machine learning-driven. The pattern matching approaches take input pre-defined pattern library of known hotspots, but they cannot detect unknown hotspots. Fortunately, detection methods which are built upon machine learning methodologies [4–9], and particularly deep learning techniques [10–14], are able to offer fast and accurate solutions to both known and unknown hotspot patterns.

Motivated by the recent advances of deep learning in other domains like computer vision, Yang *et al.* firstly proposed the hotspot detector based on a shallow convolutional neural network (CNN), where layout clips were firstly converted into the frequency domain via discrete cosine transform (DCT) before fed into networks [10]. To extract DCT features in different scales, the inception modules are introduced in the work [11] to modify the structure of neural networks. In [12], to overcome the limitation imposed by the number of labelled hotspots or non-hotspots, the concept of semi-supervised learning was introduced so that unlabeled data can be harnessed as well. By considering input layout clips as binary images, Jiang *et al.* employed a binarized neural network to further speed up the detection process [13].

These prior arts have been proven very effective, nevertheless, most of them fall into a two-stage flow, where layout feature extraction and detection process are separable. Moreover, existing techniques for layout feature extraction lack both good understandings of similarity among different layout clips and guidances from supervised information. Only a few hotspot detectors [13] ensemble the feature extraction and detection into a one-stage framework. Unfortunately, the features which are automatically learned by

ICCAD '20, November 2–5, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8026-3/20/11…$15.00
https://doi.org/10.1145/3400302.3415661

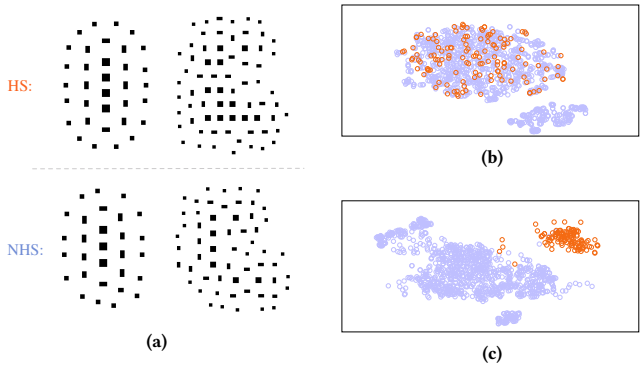

Figure 2: The snapshots of layout clips from the via layer benchmarks and Barnes-Hut t-SNE visualizations of feature embeddings on the same benchmarks: (a) The examples of hotspots and non-hotspots; (b) The DCT feature embeddings of TCAD'19; (c) The feature embeddings of our proposed framework. The via layer benchmarks consist of via patterns that contain vias and model-based sub-resolution assist features (SRAFs). Here "via" refers to the via connecting multiple metal layers.

convolutional kernels may be ill-separated owing to the lack of discrimination-oriented similarity metrics. Without an effective weighing procedure on the features per their informative significance, those redundant or even misleading features may very well degrade the overall prediction accuracy and performance. In light of these facts, those existing works can only solve simple benchmarks with apparent hotspot patterns like ICCAD 2012 Contest benchmarks, whilst in the case where non-hotspot patterns and hotspot patterns become increasing similar brings a big challenge to them. To visualize our concern, we sample some hotspots and non-hotspots from ICCAD12 [15] and new via layer benchmarks as exemplars in Figures 1(a) and 2(a). Besides, the embeddings learned from TCAD'19 and ours on both benchmark suites are projected into a two-dimension space (shown in Figures 1(b), 1(c), 2(b) and 2(c)). It can be seen that, in easy benchmarks like ICCAD12, the layout clips sharing the same label are usually similar, while the layout clips from different classes have prominent differences. Opposite to ICCAD12 benchmarks, layout clips in the via benchmarks from different classes may look very similar. What's worse, the lack of diversity of via layer patterns introduces an additional challenge to existing hotspot detectors. According to the distributions of layout feature embeddings displayed in Figures 1(b) and 2(b), we can infer that although existing works like TCAD'19 work well on ICCAD12 benchmarks, they may have poor performance on more challenging benchmarks. As demonstrated above, layout patterns in such easy benchmarks like ICCAD12 contain too much discriminative information to determine the true states of hotspot detectors.

In this work, we argue that it is of great importance to learn good embeddings of layout clips, which can be assembled into one-stage hotspot detection flow. In accordance with this argument, we propose an end-to-end detection framework where the two tasks, learning embeddings and classification, are jointly performed and mutually benefited. Furthermore, in order to focus on important features and suppressing unnecessary ones, we introduce an attention module and incorporate it into the proposed hotspot detector. To the best of our knowledge, this is the first work for layout embedding learning seamlessly combining with hotspot detection task, and there is no prior work applying attention mechanism based deep metric learning into hotspot detection. To evaluate the true efficacy of existing works and the proposed detector, we adopt a much more challenging benchmark suite. Figures 1(c) and 2(c) prove the performance of the proposed framework on both easy and more challenging benchmarks to some extent. Our main contributions are listed as follows.

- Leverage deep layout metric learning model to learn the good layout feature embeddings.
- Analyze the generalized representation ability of triplet-based deep metric learning algorithm.
- Incorporate feature embedding and hotspot detection into a one-stage multi-branch flow.
- Apply attention mechanism to learn better feature embeddings.
- A more challenging benchmark suite that fills the void created by previous easy benchmarks will be released.
- The proposed detector improves the performance on accuracy, false alarm, and runtime of inference compared to the state-of-the-art frameworks.

The rest of the paper is organized as follows. Section 2 introduces some preliminaries on metrics and problem formulation in the paper. Section 3 first gives a whole view of the proposed framework, then illustrates the backbone network with the applied inception and attention modules, as well as multi-branch design. Section 4 describes the loss functions for each branch and some training strategies. Section 5 depicts the new via benchmarks and then provides experimental results, followed by the conclusion in Section 6.

2 Problem Formulation

In general, our task can be defined as an image classification problem. Our proposed framework treats input layout clips as images. The label of a layout clip is given according to the information that whether the core region [15] of the clip contains hotspots or not.

We adopt the same metrics exploited in previous work to evaluate the performance of our proposed hotspot detector. The following show definitions of these metrics.

Definition 1 (Accuracy)**.** The ratio between the number of correctly categorized hotspot clips and the number of real hotspot clips.

Definition 2 (False Alarm)**.** The number of non-hotspot clips that are classified as hotspots by the classifier.

With the evaluation metrics above, our problem is formulated as follows.

Problem 1 (Hotspot Detection)**.** Given a collection of clips containing hotspot and non-hotspot layout patterns, the objective of deep layout metric learning-based hotspot detection is training a model to learn optimal feature embeddings and classify all the clips so that the detection accuracy is maximized whilst the false alarm is minimized with a less runtime overhead.

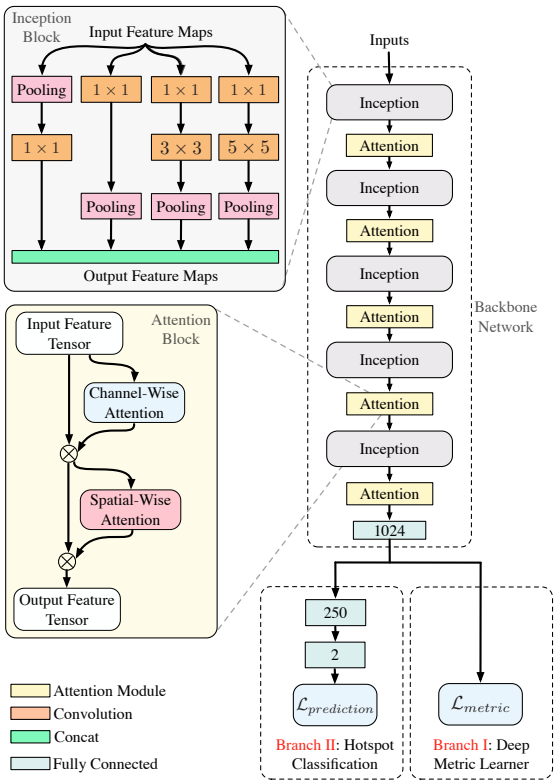

Figure 3: The architecture of the proposed hotspot detector.

3 Hotspot Detection Architecture

3.1 Overall Framework

The prior arts are either in a two-stage framework or lacking discriminative feature extractor. By contrast, our proposed algorithm adopts a well-designed multi-branch flow which works in an end-to-end manner. During the training, the proposed multi-branch flow simultaneously works on two branches: one is feature embedding and the other is classification. When the training process finishes, our hotspot detector identifies not only layout feature embeddings, but also a pretty good boundary to divide the hotspots and non-hotspots in embedding space. By the merit of the end-to-end nature, our detector is fast and flexible for both training and inference phases.

Figure 3 shows the architecture of the proposed framework, where "⊗" stands for the element-wise multiplication. The proposed framework is composed of three main components: (1) Backbone network which is based on inception structures and attention modules. It is shared by two jointly learned tasks and then is split into two branches. The whole backbone includes 5 inception modules, 5 attention modules and 1 fully connected layer; (2) For deep layout metric learner branch, it is guided by a triplet loss function to strive for good embeddings of layout clips; (3) For hotspot classification branch, it behaves as an ordinary learning model-based hotspot detector as in other works.

3.2 Backbone: Inception Block

Recent progress of deep learning techniques in computer vision reveals that by virtue of the increasing model complexity, a deeper neural network achieves a more robust feature expression and a higher accuracy comparing to a shallow one. However, deeper networks are susceptible to be overfitted, and gradient vanish emerges. What is worse, the turn-around-time at inference stage and training stage are greatly affected, too.

In our context, more cases are needed to concern. For instance, a layout pattern which usually contains several rectangles is monotonous. Another example is in our via pattern, the distances between vias and surrounding SRAFs, the distances among vias have pretty large variations. A single-sized kernel cannot capture multi-scale information. To tackle these issues, we employ an Inception-based structure [17] which consists of several convolutional kernels with multiple sizes. At the same level, these convolutional kernels perform convolution operations with different kernel scales on layout patterns, and the outputs are concatenated in the channel dimension. After going through pooling layers, the fused feature maps are scaled down in height and width dimensions. As a result, the feature maps are comprehensive and rich. The backbone network based on inception structure is illustrated in Figure 3.

3.3 Backbone: Attention Block

Recently presented attention mechanism which mimics human perception has achieved much success in the computer vision domain [18–20]. With attention techniques, neural networks focus on the salient parts of input features and generate attention-aware feature maps. In order to capture structures of feature maps better, we exploit this mechanism and embed the corresponding modules into our backbone network.

As we know, the features include both cross-channel and spatial information on the back of convolution computations. Based on that fact, the embedded attention modules can emphasize informative parts and suppress unimportant ones along the channel and spatial axes. Channel-wise attention focuses on the informative part itself, while spatial attention concentrates on its spatial location. They are complementary to each other. To fit the motivation, the structure of one attention module consists of two sub-parts: one is channel-wise, and the other is spatial-wise. For a better understanding, we visualize an attention module and zoom in on its intra-structure in Figure 3. The whole attention module sequentially infers a 1D channel attention map, and then a 2D spatial attention map. Through the broadcasting operation, each attention map will perform element-wise multiplication with input feature maps.

The whole process of channel-wise attention is concluded in Equations (1) and (2), and visualized in Figure 4. First, given an input feature tensor $\boldsymbol{T}$, spatial information of $\boldsymbol{T}$ is firstly aggregated by average-pooling and max-pooling operations, and then two different spatial context descriptors are produced. Next, two descriptors go through a shared encoder-decoder network (i.e. a single hidden layer perceptron named ED() in Equation (2)), and output feature vectors are merged using element-wise summation. After activation operation (i.e. defined by $\sigma()$ in Equation (2)), the elements of the channel-wise attention masks $A_c(\boldsymbol{T})$ are firstly broadcasted along the spatial dimension, then multiplied with corresponding elements in feature maps. Finally, the module outputs the feature

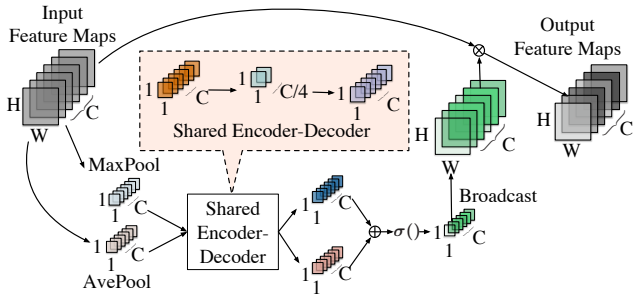

Figure 4: The channel-wise attention module.

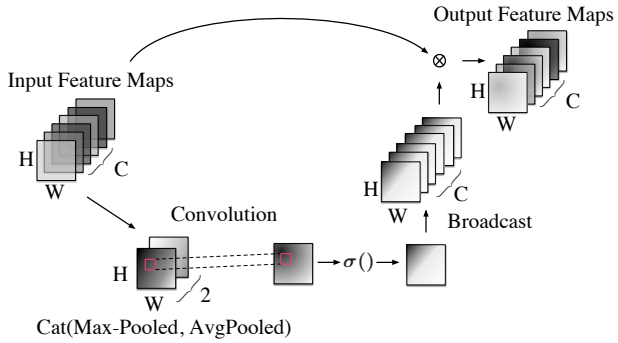

Figure 5: The spatial-wise attention module.

tensor $\boldsymbol{T}'$. In a nutshell, the channel-wise attention module infers the channel-wise attention masks, and each mask will be multiplied element-wisely with input feature maps.

$$\boldsymbol{T}' = A_c(\boldsymbol{T}) \otimes \boldsymbol{T}, \tag{1}$$

$$A_c(\boldsymbol{T}) = \sigma(\mathrm{ED}(\mathrm{AvgPool}(\boldsymbol{T})) + \mathrm{ED}(\mathrm{MaxPool}(\boldsymbol{T}))). \tag{2}$$

We mathematically summarize the computation process for spatial-wise attention module in Equations (3) and (4), and visualize it in Figure 5. First, through pooling operations, we aggregate channel information of the input feature tensor $\boldsymbol{T}'$, which is also the output of channel-wise attention module. Then, concatenate (denoted by Cat() in Equation (4)) two aggregated features and perform convolution computation (i.e. Conv()) on the concatenation. After activated by sigmoid function, the spatial attention mask $A_s(\boldsymbol{T}')$ is generated. At last, the mask will be multiplied with $\boldsymbol{T}'$ element-wisely, which is shown in Equation (3).

$$\boldsymbol{T}'' = A_s(\boldsymbol{T}') \otimes \boldsymbol{T}', \tag{3}$$

$$A_s(\boldsymbol{T}') = \sigma(\mathrm{Conv}(\mathrm{Cat}(\mathrm{AvgPool}(\boldsymbol{T}'), \mathrm{MaxPool}(\boldsymbol{T}')))). \tag{4}$$

3.4 Multi-branch Design

Our end-to-end framework has two jointly-performed tasks: hotspot classification and deep layout metric learning. The two tasks share the backbone network for feature extraction, but are guided by different loss functions. For the deep layout metric learner, the proposed triplet loss is directly calculated on the high dimensional vector which is the output of the fully connected layer in the backbone network. The learner searches for a good feature embedding which non-linearly maps the image representations of layout patterns (i.e. grey images) into a new space. Therefore, pairs of hotspot patterns and non-hotspot patterns can be effectively measured and separated by Euclidean distance metric. For hotspot detection, the main task is to find an appropriate boundary that well divides hotspots and non-hotspots. Via back-propagation, the guide information (i.e. gradients) from two branches update the backbone collectively.

4 Loss Functions and Training

4.1 Metric Learning Loss in Branch I

Metric learning is typically referred to learning a distance metric or pursuing an embedding function to map images onto a new manifold. Given a similarity metric function, similar images are projected into a neighbourhood, while dissimilar images are mapped apart from each other. Note that the term "similar images" means the images share the same label. Over past decades, the machine learning community has witnessed a remarkable growth of metric learning algorithms used in a variety of tasks such as classification [21], clustering [22], image retrieval [23] and etc. However, many metric learning approaches explore only linear correlations, thus may suffer from nonlinearities among samples. Although kernel tricks have been developed, it is resource-consuming to find an appropriate one. With a notable success achieved by deep learning, deep metric learning methods have been proposed to find the non-linear embeddings. By taking advantage of deep neural networks to learn a non-linear mapping from the original data space to the embedding space, deep metric learning methods measuring Euclidean distance in the embedding space can reflect the actual semantic distance between data points.

Contrastive loss [24] and triplet loss [25] are two conventional measures which are widely utilized in most existing deep metric learning methods. The contrastive loss is designed to separate samples of different classes with a fixed margin and pull closer samples of the same category as near as possible. The triplet loss is more effective and more complicated, which contains the triplets of anchors, positive and negative samples. Since the triplet loss takes into consideration of higher-order relationships in embedding space and thus can achieve better performance than the contrastive loss. Therefore, in the proposed deep layout metric learning, we adopt the triplet loss.

As aforementioned, the goal of deep metric learning aims at finding a good embedding which is denoted by $f_{\boldsymbol{w}}(\boldsymbol{x}) \in \mathbb{R}^d$ with $\boldsymbol{w}$ as the parameter of f in our assumption. The embedding function $f_{\boldsymbol{w}}(\cdot)$ projects the layout pattern $\boldsymbol{x}$ onto a hypersphere located in a $d-$dimensional compact Euclidean space, where distance directly corresponds to a measure of layout similarity. In other words, the normalization constraint $||f_{\boldsymbol{w}}(\boldsymbol{x})||_2^2 = 1$ is attached to all embeddings. The mechanism behind the triplet loss is based on the following rules:

- During training, the layout clips constitute several triplet instances, where $f_{\boldsymbol{w}}(\boldsymbol{x}_i)$, $f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+)$, $f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)$ denote an anchor layout clip, a layout clip sharing the same label with the anchor and a layout clip which has the opposite label, respectively;

- Each triplet instance indicates the triplet relationship among three layout clips as:

$$S(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)) + M < S(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+)),$$
$$\forall (f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)) \in \mathcal{T}. \tag{5}$$

  In Equation (5), S is the similarity measurement metric and can produce the similarity values always satisfying the triplet relationship. M refers to a margin between positive and negative pairs, and $\mathcal{T}$ collects all valid triplets in training set with $|\mathcal{T}| = n$.
- The Euclidean distance is employed as the metric to measure the similarity between the embedding pairs in new feature space.

To explore the relationship among triplet layout clips, the objective function of deep layout metric learning can be formulated as a loss function $\mathcal{L}_{metric}((f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)))$, which is based on the hinge loss (displayed in Equation (6a)) with Constraint (6b). $\mathcal{L}_{metric}((f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)))$ is also called empirical loss. Minimizing the empirical loss is equivalent to minimizing the violations on the relationship defined in Equation (5).

$$\min_{\boldsymbol{w}} \frac{1}{n}\sum_{i=1}^{n} \max(0, M + ||f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+)||_2^2 - ||f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)||_2^2) \tag{6a}$$

$$\text{s.t. } ||f_{\boldsymbol{w}}(\boldsymbol{x}_i)||_2^2 = 1,\ \forall (f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)) \in \mathcal{T}. \tag{6b}$$

The illustration for the proposed loss is shown in Figure 6. It can be seen that after training, the layout clips of the same category will be kept apart from the one which is from the other class. The proposed layout triplet loss attempts to enforce a margin between each pair of layout clips from hotspot to non-hotspot. To prove the property, we calculate the gradients of $\mathcal{L}_{metric}$ with respect to the embedding vectors by the following Equations (7a) to (7c).

$$\frac{\partial \mathcal{L}_{metric}(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-))}{\partial f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+)} = \frac{2}{n}\left(f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+) - f_{\boldsymbol{w}}(\boldsymbol{x}_i)\right) \cdot \mathbf{1}\left(\mathcal{L}_{metric}(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)) > 0\right), \tag{7a}$$

$$\frac{\partial \mathcal{L}_{metric}(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-))}{\partial f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)} = \frac{2}{n}\left(f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)\right) \cdot \mathbf{1}\left(\mathcal{L}_{metric}(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)) > 0\right), \tag{7b}$$

$$\frac{\partial \mathcal{L}_{metric}(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-))}{\partial f_{\boldsymbol{w}}(\boldsymbol{x}_i)} = \frac{2}{n}\left(f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-) - f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+)\right) \cdot \mathbf{1}\left(\mathcal{L}_{metric}(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)) > 0\right), \tag{7c}$$

where $\mathbf{1}$ is the indicator function which is defined as:

$$\mathbf{1}(x) = \begin{cases} 1 & \text{if x is true,} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

When the distance constraint is satisfied, i.e. there are no violations on the triplet relationship, gradients become zeros. As a result, triplet loss $\mathcal{L}_{metric}$ allows the layouts from one category to live on a manifold, while still keep the distance and hence discriminative to another category.

We further analyze the effectiveness of triplet loss by offering its bias between the generalization error and the empirical error.

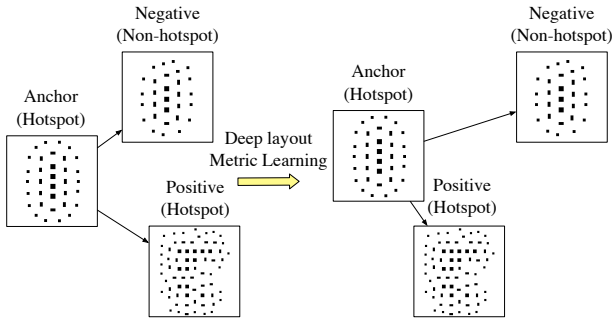

Figure 6: The visualization of the proposed deep layout metric learning. In the worst case, the anchor is much similar to the negative than to the positive. In other words, in original space, the distance between the anchor and the negative is shorter than the one between the anchor and the positive. But after deep layout metric learning, in a new manifold, the two hotspot layout clips are kept apart from the non-hotspot clip.

In other words, we evaluate the upper-bound of the network representation ability in a more mathematical way. In the following descriptions, for a better explanation, we use $(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k)$ expressing a valid triplet. Before showing the bias, we readily extend the pair-based definitions in [26] to the triplet scenarios, as in Lemma 1.

Lemma 1. *A triplet-based metric learning algorithm has β-uniform stability ($\beta \geq 0$), if*

$$\sup_{\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k \sim \mathcal{D}} \left| \ell\left(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot), \boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k\right) - \ell\left(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot), \boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k\right) \right| \leq \beta, \quad \forall \mathcal{T}, \mathcal{T}_i, \tag{9}$$

where ℓ indicates a loss function, $\mathcal{T}_i$ is the training set $\mathcal{T}$ with sample $\boldsymbol{x}_i$ replaced by an independent and identically distributed exemplar $\boldsymbol{x}_i'$, and $\mathcal{D}$ is some kind of distribution. $f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)$ and $f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot)$ are mapping functions learned over the training set $\mathcal{T}$ and $\mathcal{T}_i$, respectively.

Theorem 1. *Assume ℓ be a loss function upper-bounded by $\mathcal{B} \geq 0$, and let $\mathcal{T}$ be a training set consisting of n valid triplets drawn from distribution $\mathcal{D}$, and $f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)$ the mapping function parameterized by $\boldsymbol{w}$ which is learned over the training set $\mathcal{T}$ by a β-uniformly stable deep metric learner. The empirical loss over $\mathcal{T}$ is $\mathcal{L}(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot))$, while the expected loss of learned mapping function $f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)$ over distribution $\mathcal{D}$ is $\mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k \sim \mathcal{D}}[\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot), \boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k)]$. Then, for $0 < \delta < 1$, with confidence $1 - \delta$ approaching to 1, the following inequality exists:*

$$\mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k \sim \mathcal{D}}[\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot), \boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k)] - \mathcal{L}(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)) \leq 3\beta + (2n\beta + 3\mathcal{B})\sqrt{\frac{\log \frac{1}{\delta}}{2n}}. \tag{10}$$

The proof of Theorem 1 is detailed in Appendix A. It can be observed that with the increasing of the number of training triplets, the bias converges. Our analyses for generalized representation ability of triplet-based deep metric learning demonstrate the performance of the proposed layout metric learner.

4.2 Classification Loss in Branch II

Except for the same backbone network, there are two fully-connected layers in our classification branch. Different from those

hotspot detectors in previous deep learning-based work, the proposed classifier predicts labels based on the features learned by backbone network and deep layout metric learner. Like prior art [10], the loss function for classification, defined by $\mathcal{L}_{prediction}$, is cross-entropy loss:

$$-(y\log(y^*)+(1-y)\log(1-y^*)), \tag{11}$$

where y^* is the predicted probability of a layout clip, while y is the relevant ground truth (binary indicator).

4.3 Training Strategies

Hotspot detection is haunted with a crucial issue that relative datasets (e.g. ICCAD12 benchmark suite [15]), no matter in academia or industry, are quite unbalanced. That is, the number of hotspots is much less than that of non-hotspots. This property results in a biased classification towards the non-hotspots. Additionally, limited by the bottleneck of hardware, it is infeasible to compute the arg min or arg max across the whole training set. Hence, sampling matters in our framework.

What we do is to generate triplets from a balanced mini-batch in an online fashion. This can be done by constructing a balanced mini-batch (i.e. the numbers of hotspots and non-hotspots are equal) in one iteration and then selecting the hard negative exemplars from within the batch. Here we divide the negative samples based on a simple rule that easy negatives will lead the loss to become zero, whilst the hard negatives make the loss valid. Since only picking the hardest negatives leads to bad local minima early during training, we keep all anchor-positive pairs in a mini-batch while selectivity sample some hard negatives. Besides hardest negatives, we also consider some negative exemplars which are further away from the anchor than the positive samples but still hard. Those so-called "semi-hard" negatives which lie inside the margin obey the following inequality:

$$\begin{aligned}||f_{\boldsymbol{w}}(\boldsymbol{x}_i)-f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+)||_2{}^2 &\le ||f_{\boldsymbol{w}}(\boldsymbol{x}_i)-f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)||_2^2\\ &\le M+||f_{\boldsymbol{w}}(\boldsymbol{x}_i)-f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+)||_2^2.\end{aligned} \tag{12}$$

Aiming at progressively selecting false-positive samples that will benefit training, this kind of sampling strategy is widely used in deep metric learning methodologies. Many visual tasks [27–31] have proven its effectiveness. One reason is that it reduces the number of layout tuples that can be formed for training, and thus enhances the training efficiency.

Nonetheless, the problem that the hotspot clip number is much less than the non-hotspot clip number still exists. We perform top-bottom and left-right flipping on hotspot clips, as flipping a clip does not affect its property during lithography process [10]. This data augmentation will introduce diversity to the dataset and further increase the generalization of our model.

5 Experiment Results

The implementation of our framework is in `Python` with the `TensorFlow` library [32], and we test it on a platform with the Xeon Silver 4114 CPU processor and Nvidia TITAN Xp Graphic card. To verify the effectiveness and efficiency of our detector, two benchmarks are employed. One is ICCAD12 benchmarks [15], and the other is a more challenging via layer benchmark suite which is under 45*nm* technology node. For fair comparisons against previous works, following these arts, all 28*nm* designs in ICCAD12 benchmarks [15] are merged into a unified case named `ICCAD12`. The details for `ICCAD12` and the via benchmarks are listed in Table 1.

Table 1: Benchmark Statistics

Benchmarks	Training Set		Testing Set		Size/Clip (μm^2)
	HS#	NHS#	HS#	NHS#	
ICCAD12	1204	17096	2524	13503	3.6×3.6
Via-1	3418	10302	2267	6878	2.0×2.0
Via-2	1029	11319	724	7489	2.0×2.0
Via-3	614	19034	432	12614	2.0×2.0
Via-4	39	23010	26	15313	2.0×2.0
Via-Merge	5100	63665	3449	42294	2.0×2.0

Columns "Train HS#" and "Train NHS#" list the total number of hotspots and the total number of non-hotspots in the training set, whilst columns "Test HS#" and "Test NHS#" refer to the total number of hotspots and the total number of non-hotspots in the testing set. "Size/Clip (μm^2)" shows the resolution for each benchmark. It is manifest that the via benchmark suite has four individual cases, which are arranged in order of the density of target designs. For example, `Via-1` has the highest density of target designs in its layout clips among other cases. As a result, it contains the most hotspot patterns. In a low-density case, like `Via-4`, the number of hotspot clips are reduced accordingly. We also merge all four small cases into a big one named "`Via-Merge`". Note that, as listed in Table 1, images in the testing set of `ICCAD12` have a resolution of 3600×3600 which is larger than the images in the via benchmarks of 2048×2048.

Table 2 summarizes the comparing results between the proposed framework and several state-of-the-art hotspot detectors. Column "Bench" lists 6 benchmarks used in our experiments. Columns "Accu", "FA", "Time" are hotspot detection accuracy, false alarm count and detection runtime, respectively. Columns "TCAD'19 [10]", "DAC'19 [13]", "ASPDAC'19 [12]" and "JM3'19 [11]" denote the results of selected baseline frameworks respectively. We can see that our framework outperforms TCAD'19 averagely with 15.22% improvement on detection accuracy and 2% less false alarm penalty. Especially, our framework behaves much better on average with 89.89% detection accuracy compared to 77.78%, 83.06%, 58.93% and 62.9% for TCAD'19, DAC'19, ASPDAC'19, JM3'19, respectively. Moreover, the advantage of the proposed one-stage multi-branch flow can also be noticed that it achieves over 3× speedup compared to previous two-stage flows. Note that DAC'19 exhibits a slightly better accuracy on `ICCAD12` case, it suffers from high false alarm penalties over almost all cases due to the nature of the binarized neural network.

An ablation study is also performed on the proposed flow to investigate how different configurations affect the performance. Figure 7 illustrates the contribution of attention module, inception block and layout metric learning loss to our flow. "w/o. Atten" refers to the detector without attention modules, "w/o. Incept" stands for the detector with inception blocks replaced by vanilla convolutional layers, "w/o. Metric" denotes the detector trained without layout

Table 2: Comparison with state of the arts

Bench	TCAD'19 [10]			DAC'19 [13]			ASPDAC'19 [12]			JM3'19 [11]			Ours		
	Accu (%)	FA	Time (s)	Accu (%)	FA	Time (s)	Accu (%)	FA	Time (s)	Accu (%)	FA	Time (s)	Accu (%)	FA	Time (s)
ICCAD12	98.40	3535	502.70	98.54	3260	561.28	97.66	2825	441.96	97.82	2651	505.67	98.42	2481	143.79
Via-1	71.50	773	43.36	89.85	1886	57.76	64.18	1077	52.95	89.19	2624	47.87	93.42	1589	19.83
Via-2	65.06	1290	40.02	73.00	1222	21.66	30.52	372	43.21	38.81	454	43.06	86.32	1100	13.22
Via-3	48.15	760	60.23	73.38	3406	43.15	26.92	148	77.09	21.06	42	67.13	88.20	2105	20.69
Via-4	76.92	155	67.44	73.08	15288	51.98	61.54	74	87.24	46.15	21	77.15	80.77	152	20.70
Via-Merge	88.01	7633	165.85	90.42	9295	105.30	72.77	3859	228.57	84.34	6759	170.91	92.20	6453	59.74
Average	74.67	2357.67	146.60	83.06	5726.17	140.19	58.93	1392.50	155.17	62.90	2091.83	151.97	89.89	2313.33	46.33
Ratio	0.83	1.02	3.16	0.92	2.48	3.03	0.66	**0.60**	3.35	0.70	0.90	3.28	**1.00**	1.00	**1.00**

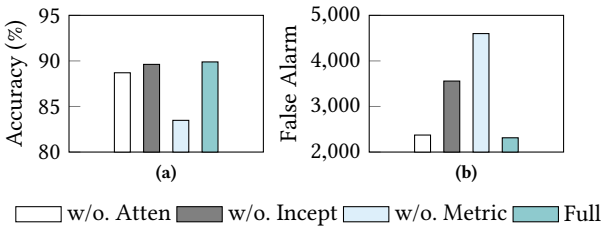

Figure 7: Comparison among different configurations on (a) average accuracy and (b) average false alarm.

metric learning loss, whilst "Full" is our detector with entire techniques. The histogram shows that with attention modules, 1.29% accuracy improvement without additional false alarms on average is achieved, which confirms that the attention modules help the backbone network extract feature more efficiently. With inception blocks, we get a notable reduction on false alarm penalties, i.e. 1245 less on average, which means under the same experiment settings, the inception blocks capture richer information on layout patterns than vanilla convolutional layers. Comparing the whole framework with the model trained without layout metric learning, the model trained with layout metric learning loss reduces 49.72% of the false alarm and get 6.41% further improvement on accuracy.

6 Conclusion

In this paper, for the first time, we have proposed a new end-to-end hotspot detection flow where layout feature embedding and hotspot detection are simultaneously performed. The deep layout metric learning mechanism offers a new solution to extract features from via layer patterns that contain much less discriminative information than metal layer patterns. We further exploit attention techniques to make backbone network self-adaptively emphasize on more informative parts. Additionally, to test the true performance of hotspot detectors, a new via layer benchmark suite has been used for comprehensive verification. The experimental results demonstrate the superiority of our framework over current deep learning-based detectors. The corresponding theoretical analyses are also provided as the pillars. With the transistor size shrinking rapidly and the layouts becoming more and more complicated, we hope to apply our ideas into more general VLSI layout feature learning and encoding.

7 Acknowledgment

This work is partially supported by The Research Grants Council of Hong Kong SAR (No. CUHK14209420), and National Natural Science Foundation of China (NSFC) research projects 61822402, 61774045, 61929102 and 62011530132.

References

[1] S.-Y. Lin, J.-Y. Chen, J.-C. Li, W.-Y. Wen, and S.-C. Chang, "A novel fuzzy matching model for lithography hotspot detection," in *Proc. DAC*, 2013, pp. 68:1–68:6.
[2] W.-Y. Wen, J.-C. Li, S.-Y. Lin, J.-Y. Chen, and S.-C. Chang, "A fuzzy-matching model with grid reduction for lithography hotspot detection," *IEEE TCAD*, vol. 33, no. 11, pp. 1671–1680, 2014.
[3] S. S.-E. Tseng, W.-C. Chang, I. H.-R. Jiang, J. Zhu, and J. P. Shiely, "Efficient search of layout hotspot patterns for matching sem images using multilevel pixelation," in *Proc. SPIE*, vol. 10961, 2019, p. 109610B.
[4] D. Ding, B. Yu, J. Ghosh, and D. Z. Pan, "EPIC: Efficient prediction of IC manufacturing hotspots with a unified meta-classification formulation," in *Proc. ASPDAC*, 2012, pp. 263–270.
[5] B. Yu, J.-R. Gao, D. Ding, X. Zeng, and D. Z. Pan, "Accurate lithography hotspot detection based on principal component analysis-support vector machine classifier with hierarchical data clustering," *JM3*, vol. 14, no. 1, p. 011003, 2015.
[6] T. Matsunawa, J.-R. Gao, B. Yu, and D. Z. Pan, "A new lithography hotspot detection framework based on AdaBoost classifier and simplified feature extraction," in *Proc. SPIE*, vol. 9427, 2015.
[7] H. Zhang, B. Yu, and E. F. Y. Young, "Enabling online learning in lithography hotspot detection with information-theoretic feature optimization," in *Proc. ICCAD*, 2016, pp. 47:1–47:8.
[8] H. Geng, H. Yang, B. Yu, X. Li, and X. Zeng, "Sparse VLSI layout feature extraction: A dictionary learning approach," in *Proc. ISVLSI*, 2018, pp. 488–493.
[9] W. Ye, M. B. Alawieh, M. Li, Y. Lin, and D. Z. Pan, "Litho-gpa: Gaussian process assurance for lithography hotspot detection," in *Proc. DATE*, 2019, pp. 54–59.
[10] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Y. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE TCAD*, vol. 38, no. 6, pp. 1175–1187, 2019.
[11] J. Chen, Y. Lin, Y. Guo, M. Zhang, M. B. Alawieh, and D. Z. Pan, "Lithography hotspot detection using a double inception module architecture," *JM3*, vol. 18, no. 1, p. 013507, 2019.
[12] Y. Chen, Y. Lin, T. Gai, Y. Su, Y. Wei, and D. Z. Pan, "Semi-supervised hotspot detection with self-paced multi-task learning," in *Proc. ASPDAC*, 2019, pp. 420–425.
[13] Y. Jiang, F. Yang, H. Zhu, B. Yu, D. Zhou, and X. Zeng, "Efficient layout hotspot detection via binarized residual neural network," in *Proc. DAC*, 2019, p. 147.
[14] R. Chen, W. Zhong, H. Yang, H. Geng, X. Zeng, and B. Yu, "Faster region-based hotspot detection," in *Proc. DAC*, 2019, pp. 1–6.
[15] A. J. Torres, "ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite," in *Proc. ICCAD*, 2012, pp. 349–350.
[16] L. Van Der Maaten, "Accelerating t-sne using tree-based algorithms," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3221–3245, 2014.
[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. CVPR*, 2016, pp. 2818–2826.

[18] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *cvpr*, 2017, pp. 3156–3164.
[19] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "Cbam: Convolutional block attention module," in *eccv*, 2018, pp. 3–19.
[20] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *cvpr*, 2019, pp. 3146–3154.
[21] Y. Cui, F. Zhou, Y. Lin, and S. Belongie, "Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop," in *Proc. CVPR*, 2016, pp. 1153–1162.
[22] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *Proc. NIPS*, 2003, pp. 521–528.
[23] X. Gao, S. C. Hoi, Y. Zhang, J. Wan, and J. Li, "Soml: Sparse online metric learning with application to image retrieval," in *Proc. AAAI*, 2014, pp. 1206–1212.
[24] J. Hu, J. Lu, and Y.-P. Tan, "Discriminative deep metric learning for face verification in the wild," in *Proc. CVPR*, 2014, pp. 1875–1882.
[25] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng, "Person re-identification by multi-channel parts-based cnn with improved triplet loss function," in *Proc. CVPR*, 2016, pp. 1335–1344.
[26] R. Jin, S. Wang, and Y. Zhou, "Regularized distance metric learning: Theory and algorithm," in *Proc. NIPS*, 2009, pp. 862–870.
[27] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. CVPR*, 2015, pp. 815–823.
[28] C. Huang, C. C. Loy, and X. Tang, "Local similarity-aware deep feature embedding," in *Proc. NIPS*, 2016, pp. 1262–1270.
[29] Y. Yuan, K. Yang, and C. Zhang, "Hard-aware deeply cascaded embedding," in *Proc. ICCV*, 2017, pp. 814–823.
[30] B. Harwood, B. Kumar, G. Carneiro, I. Reid, T. Drummond *et al.*, "Smart mining for deep metric learning," in *Proc. ICCV*, 2017, pp. 2821–2829.
[31] R. Yu, Z. Dou, S. Bai, Z. Zhang, Y. Xu, and X. Bai, "Hard-aware point-to-set deep metric for person re-identification," in *Proc. ECCV*, 2018, pp. 188–204.
[32] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. OSDI*, 2016, pp. 265–283.
[33] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.
[34] B. Wang, H. Zhang, P. Liu, Z. Shen, and J. Pineau, "Multitask metric learning: Theory and algorithm," in *Proc. AISTATS*, 2019, pp. 3362–3371.
[35] C. McDiarmid, "On the method of bounded differences," *Surveys in combinatorics*, vol. 141, no. 1, pp. 148–188, 1989.

A The Proof for Theorem 1

Proof. The proof follows from [33, 34]. $F_{\mathcal{T}}$ is defined as a replacement to $\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)]-\mathcal{L}(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot))$. From Equation (13a) to Equation (13b), we exploit the triangle inequality. Then, the upper-bound of Equation (13b) is attained by using Jensen's inequality and the definition of $\mathcal{L}$. With the combination of the triangle inequality, β-uniform stability and $\mathcal{B}$-boundedness, the further bound is found in Equation (13d).

Based on the upper-bound obtained in Equation (13), we utilize McDiarmid's inequality [35] to obtain Equation (14).

$$\Pr[F_{\mathcal{T}}\geq\epsilon+\mathbb{E}[F_{\mathcal{T}}]]\leq\exp\left(\frac{-2n\epsilon^2}{(2n\beta+3B)^2}\right). \tag{14}$$

With δ set to be $\exp\left(\frac{-2n\epsilon^2}{(2n\beta+3\mathcal{B})^2}\right)$, ϵ equals to $(2n\beta+3\mathcal{B})\sqrt{\frac{\log\frac{1}{\delta}}{2n}}$. Hence, with confidence $1-\delta$, Equation (15) exists.

$$F_{\mathcal{T}}\leq\mathbb{E}[F_{\mathcal{T}}]+(2n\beta+3\mathcal{B})\sqrt{\frac{\log\frac{1}{\delta}}{2n}}. \tag{15}$$

For an effective bound, $\beta=o\left(\frac{1}{\sqrt{n}}\right)$. Assume $\beta=\mathcal{O}\left(n^p\right)$. Since $\lim_{n\to+\infty}\frac{-n}{(2n\beta+3\mathcal{B})^2}=-\infty$, $1>2*(1+p)$ holds and $\beta=o\left(\frac{1}{\sqrt{n}}\right)$.

Equation (16) shows the searching computing of the upper-bound of $\mathbb{E}[F_{\mathcal{T}}]$. Note that from Equation (16a) to Equation (16b), we harness a fact that replacing the examples with i.i.d exemplars does not change the expected computation. More specifically, $\mathbb{E}_{\mathcal{T}\sim\mathcal{D}}[\mathcal{L}\left(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)\right)]=\mathbb{E}_{\mathcal{T}\sim\mathcal{D}}[\mathcal{L}\left(f_{\boldsymbol{w}_{\mathcal{T}_{i,j,k}}}(\cdot)\right)]$.

$$\begin{aligned}
&\left|F_{\mathcal{T}}-F_{\mathcal{T}_i}\right|\\
=&\left|\mathcal{L}(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot))-\mathcal{L}(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot))+\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)]\right.\\
&\left.-\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[\ell(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)]\right| &\text{(13a)}\\
\leq&\left|\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)]\right.\\
&-\left.\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[\ell(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)|\right.\\
&+\left|\mathcal{L}(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot))-\mathcal{L}(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot))\right| &\text{(13b)}\\
\leq&\beta+\frac{1}{|\mathcal{T}|}\left|\sum_{j\neq i}\sum_{k\neq i}\sum_{l\neq i}(\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_k,\boldsymbol{x}_l)-\ell(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_k,\boldsymbol{x}_l))\right.\\
&+\sum_{j\neq i}\sum_{k\neq i}(\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_k,\boldsymbol{x}_i)-\ell(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_k,\boldsymbol{x}_i'))\\
&+\sum_{j\neq i}\sum_{l\neq i}(\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_i,\boldsymbol{x}_l)-\ell(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_i',\boldsymbol{x}_l))\\
&+\left.\sum_{k\neq i}\sum_{l\neq i}(\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_k,\boldsymbol{x}_l)-\ell(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i',\boldsymbol{x}_k,\boldsymbol{x}_l))\right| &\text{(13c)}\\
\leq&\beta+\beta+\frac{3\mathcal{B}}{n}\leq 2\beta+\frac{3\mathcal{B}}{n}. &\text{(13d)}
\end{aligned}$$

$$\mathbb{E}[F_{\mathcal{T}}]=\mathbb{E}\left[\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)]-\mathcal{L}\left(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)\right)\right] \tag{16a}$$

$$\begin{aligned}
&\leq\mathbb{E}_{\mathcal{T},\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[|\ell\left(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)\\
&-\frac{1}{n}\sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)\in\mathcal{T}}\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_{i,j,k}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)|] &\text{(16b)}\\
&=\mathbb{E}_{\mathcal{T},\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}\left[\left|\frac{1}{n}\sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)\in\mathcal{T}}\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_{i,j,k}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)\right.\right.\\
&-\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_{i,j}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)+\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_{i,j}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)-\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)\\
&\left.\left.+\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)-\ell\left(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)\right|\right] &\text{(16c)}\\
&\leq\mathbb{E}_{\mathcal{T},\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}\left[\frac{1}{n}\sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)\in\mathcal{T}}\left|\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_{i,j,k}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)\right.\right.\\
&-\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_{i,j}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)\Big|+\left|\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_{i,j}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)-\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)\right|\\
&\left.+\left|\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)-\ell\left(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)\right|\right]\leq 3\beta. &\text{(16d)}
\end{aligned}$$

In Equation (16), $f_{\boldsymbol{w}_{\mathcal{T}_{i,j,k}}}(\cdot)$ is the mapping function learned over the training set $\mathcal{T}$ with $\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k$ replaced by $\boldsymbol{x}_i',\boldsymbol{x}_j',\boldsymbol{x}_k'$. Combing the results of Equation (15) and Equation (16), Inequality (10) holds. Therefore, with $\beta=o\left(\frac{1}{\sqrt{n}}\right)$, the generalization gap will converge in the order of $\mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$ with high confidence $1-\delta$. The proof completes. □