Vicinal Counting Networks

Viresh Ranjan[1] Minh Hoai[1,2]
[1]Stony Brook University, USA
[2]VinAI Research, Hanoi, Vietnam

Abstract

We tackle the task of Few-Shot Counting. Given an image containing multiple objects of a novel visual category and few exemplar bounding boxes depicting the visual category of interest, we want to count all of the instances of the desired visual category in the image. A key challenge in building an accurate few-shot visual counter is the scarcity of annotated training data due to the laborious effort needed for collecting and annotating the data. To address this challenge, we propose Vicinal Counting Networks, which learn to augment the existing training data along with learning to count. A Vicinal Counting Network consists of a generator and a counting network. The generator takes as input an image along with a random noise vector and generates an augmented version of the input image. The counting network learns to count the objects in the original and augmented images. The training signal for the generator comes from the counting loss of the counting network, and the generator aims to synthesize images which result in a small counting loss. Unlike GANs which are trained in an adversarial setting, Vicinal Counting Networks are trained in a cooperative setting where the generator aims to help the counting network in achieving accurate predictions on the synthesized images. We also show that our proposed data augmentation framework can be extended to other counting tasks like crowd counting.

1. Introduction

Visual Counting is the task of predicting the number of instances of a desired category present in an image, and it is an important computer vision problem with a wide range of applications such as crowd monitoring, traffic management, inventory management, and wildlife population tracking. Most of the contemporary visual counters [19,25,27,28,30,40,47] handle a single visual category at a time, and it is challenging to adapt these counters to new categories due to the need for training data which is expensive to obtain. In this work, we are interested in building a general visual counter capable of counting objects of many visual categories. Our work is inspired by the recent work of [29] that builds a few-shot visual counter which can generalize to novel classes using only a few examples. In this few-shot setting, as shown in Fig. 1, the visual counter takes as input an image containing multiple instances of a novel object category, and few bounding boxes from the same image depicting the object category, and predicts the number of instances of the novel object category present in the image. Such a few-shot visual counter is based on the assumption that intra-class variation within an image is not too extreme, and it can generalize to any novel visual category using only a few bounding box examples.

Although few-shot counting is a promising approach for counting objects from many visual categories, the performance of a few-shot counting network is still bounded by the limited amount of training data. Currently, the largest dataset for this task is FSC147 [29] with only 3,659 training images. To alleviate the limited size of the training set, we propose Vicinal Counting Networks (VCNs) that jointly learn to count and augment the existing training data. VCNs are inspired by the Vicinal Risk Minimization [6] (VRM) principle, which is a risk minimization principle which defines a distribution in the vicinity of data points in the training set, and samples virtual examples from this vicinity distribution to augment the training set. In recent years, VRM has inspired data augmentation strategies like Mixup [46] and Manifold Mixup [37] which perform linear combination of training examples and their corresponding labels to create virtual training examples. Mixup performs the linear interpolation in the input space, while Manifold Mixup performs linear interpolation of hidden layer representations of training examples. The VRM principle also encompasses other commonly used dataset augmentation strategies like random rotation, translation, and scaling. All these augmentation strategies rely on some form of prior knowledge. For instance, Mixup assumes that linear combination of input samples leads to a corresponding linear combination of associated targets, and random rotation assumes that rotating an input image leaves the target label unchanged. Unlike these augmentation strategies which rely on some prior knowledge, VCNs learn the augmentation strategy from the data itself, in a task dependent fashion. A VCN consists of a generator and a counting network. The generator synthe-

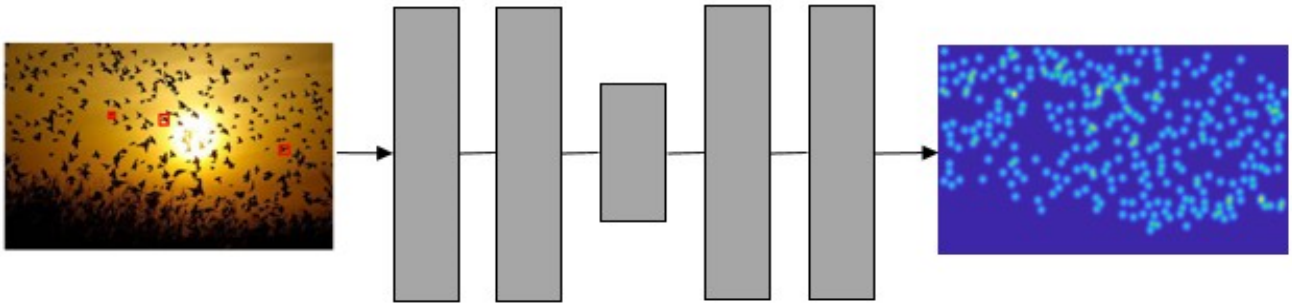

Figure 1. **Few-shot Counting.** Given an image containing multiple instances of a novel class and few examples of the class to be counted (shown in red), few-shot counting requires predicting the overall count for the novel class in the image. We present a density map estimation based approach for few-shot counting, and the overall count for the novel class is obtained by summing all of the values in the density map.

sizes images from the vicinity of training examples, and the counting network learns to count the objects in the original and synthesized virtual images. The generator effectively increases the training set size for the counting network, which helps the counting network in generalizing better. The generator takes as input an image along with a random noise vector, and generates a new image so that the counting network achieves a low error on the generated image. The entire system is trained jointly in a co-operative setting. Note that unlike GANs [10] whose training involves a minimax optimization problem which are difficult to tackle, training VCNs involve solving a typical minimization problem.

We show the effectiveness of the proposed VCNs for few-shot counting task by conducting extensive experiments on the FSC147 dataset, and improve on the previous state-of-the-art results by a significant margin. We further show that our data augmentation approach can also provide benefits in other tasks like crowd counting, and lead to state-of-the-art performance on three benchmark crowd counting datasets. Although the main focus of our work is on few-shot counting, our method can be extended to other pixel level prediction tasks like semantic segmentation, instance segmentation etc by replacing the task dependent counting network and the counting loss with an appropriate task dependent network and corresponding loss.

2. Related Works

In this work, we are interested in the task of few-shot visual counting. A lot of previous works have tackled the task of visual counting, but most of them are not developed for few-shot counting. Instead, most of the previous works treat counting as a fully supervised regression problem, and handle a single visual category at a time such as human crowd [1, 4, 5, 19, 22, 23, 25, 33, 35, 39, 42, 45, 47], cars [26], animals [3], and cells [2, 16, 44]. Most of these approaches predict a density map given an input image, and the value at each location in the density map corresponds to the density of the objects of interest present at the corresponding location in the input image. The overall count can be obtained by summing across all the locations in the density map. It is difficult to scale these approaches to handle a large number of visual categories since these approaches require a large number of dot annotations for each new visual category.

There are few methods for few-shot counting. [24] present Generic Matching Network (GMN) which is a class-agnostic visual counting approach that requires relatively less labeled data to generalize to novel categories. However, GMN still requires a few dozens to hundreds of examples to generalize to novel categories, and does not perform well if only a few examples are provided [29]. More related to ours is FamNet [29], which is a few-shot visual counter that can generalize to novel categories given only a few examples. However, the largest existing few-shot counting dataset [29] on which few-shot visual counters like FamNet can be trained consists of only a few thousand training images which limits the performance of the existing few-shot counters. Our work addresses this limited training data issue by learning to generate samples from the vicinity of training samples.

In theory, existing few-shot detectors [8, 14] can be used for few-shot counting as well. These few-shot detectors can learn to detect novel classes given a few bounding box examples, and the count can be obtained as the number of detections for an image. However, it is known that the detectors perform poorly for images with small objects and images with high levels of object density and occlusion [27]. Density estimation based approaches [29, 47] are better suited for such scenarios.

Also related to our work is the task of few-shot image classification [9, 17, 18, 31, 34, 38] which involves classifying novel visual categories based on only a few labeled instances of these novel categories. Although task and model

independent few-shot approach like MAML [9] can be applied for few-shot counting task [29], most of the few-shot image classification approaches are not suitable for our pixel level prediction task of few-shot counting.

Data augmentation has been handled by previous works [37, 46], and these strategies help in achieving better performance on image classification task. However, as will be shown by our experiments, these strategies do not perform as well when it comes the pixel level prediction task of few-shot counting.

3. Proposed Approach

In this section, we will describe the proposed Vicinal Counting Networks (VCNs) for few-shot counting. VCNs consist of a generator and a task dependent regressor network suitable for the downstream task of few-shot counting. The generator augments the set of labeled training data, and the regressor is trained on the augmented training set. Note that the generator is used only at the training time. At the test time, the generator is discarded and the regressor predicts the count for the original test image. The architecture of VCN is described in Sec. 3.1, and the training objective is discussed in Sec. 3.2. Further training details are provided in Sec. 3.3.

3.1. VCN architecture

VCNs are inspired by the Vicinal Risk Minimization principle [6], and the more recent data augmentation strategies such as Mixup [46] and Manifold Mixup [37], which aim to draw samples from the vicinity of the training samples to augment the training set. VCNs, as shown in Fig. 2, consist of two key components: the generator and the task dependent regressor, which are described next.

3.1.1 The generator network

The goal of the generator is to augment the training data by drawing virtual samples from the vicinity of a training sample so that the virtual samples adhere to the labels of the training sample. The inputs to the generator are an input image $X \in \Re^{H \times W \times 3}$ and an m-dimensional noise vector $z \in \Re^m$, and the output is an augmented version X_z of the input image.

The generator consists of a *mapping network*, an *encoder*, an *adaptive instance normalization layer*, and a *decoder*. The mapping network consists of an 8 layer MLP, similar to the style mapping network used in Style GAN [15], and maps the random noise vector z to an intermediate latent code $h(z) = w = (w^s, w^b)$. The encoder transforms the input image X into an intermediate feature representation Q by passing X through a set of convolution layers. The latent code w is used to transform the first and second order statistics of Q by means of an Adaptive Instance Normalization (AdaIN) operation [12]. Suppose Q contains K feature maps, $Q = (q_1, q_2, \dots q_i, \dots q_K)$, where q_i denotes the i^{th} feature map. AdaIN is defined as

$$AdaIN(q_i, w) = w_i^s \frac{q_i - \mu(q_i)}{\sigma(q_i)} + w_i^b. \quad (1)$$

where the feature map q_i is first normalized using the mean $\mu(q_i)$ and standard deviation $\sigma(q_i)$ of the feature map, and then scaled and translated using the *i-th* components of latent vectors w_i^s and w_i^b. Borrowing from the style-transfer literature [12], one can think of the AdaIN operation here as performing style transfer on the input features Q using the latent vector w. AdaIN preserves the content of input features, and changes its style based on the latent vector w. This results in feature map Q_w, which is mapped back into the image space by the decoder. The decoder consists of 4 residual blocks followed by 4 transposed convolution layers. The transposed convolution layers upsample the feature map. Finally a convolutional layer with 3 channels maps the upsampled feature map back into the image space. To the best of our knowledge, we are the first to present a randomized style transfer based generator for augmenting the set of labeled training data.

3.1.2 The regressor network

The regressor network f takes as input the image X and a set of bounding box locations B within X depicting the exemplar objects, and predicts the density map F, i.e., $F = f(X, B)$. Each location in the density map contains the estimated density of the objects of interest present at the corresponding location in the image, and the overall count for the objects of interest can be obtained by summing across all the locations in F. For all of our experiments, we use three exemplars per image and B contains the top-right and bottom-left coordinates of these exemplars.

The regressor network consists of three key modules: a feature representation module, a feature correlation module, and a density prediction module. The feature representation module extracts features for both the input image and the exemplars. The feature correlation module correlates the features of the exemplars with the image features. Finally, the density prediction module uses the correlation maps between the exemplars and the image features to predict the density map F. The regressor network is similar to the FamNet architecture [29] with a few necessary changes in the density prediction module to facilitate the training of the generator. Next we describe each of the three modules in details.

The Feature Representation module extracts a multi-scale convolutional feature representation of the input image and the exemplars. We use the first four convolutional

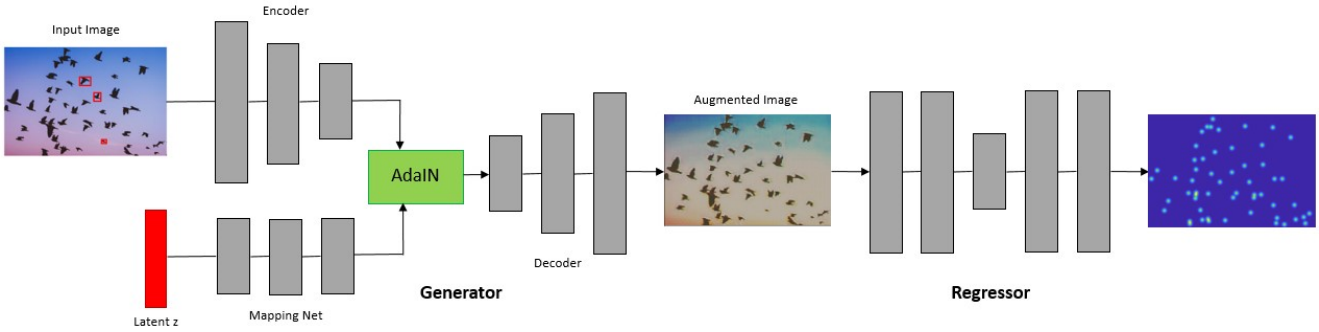

Figure 2. **Vicinal Counting Net** consists of a generator which augments the training data, and a regressor which learns to predict the density map. The count can be obtained by summing all the values in the density map. The regressor is trained on the original and synthesized images. The generator is used only during training.

blocks of the Resnet-50 architecture for the feature representation, and use the features from the third and the fourth convolutional blocks for obtaining the multi-scale representation. The multi-scale features for the exemplars are obtained by performing ROI pooling using the set of bounding boxes B as the regions of interest.

The Feature Correlation module correlates the features for each exemplar with the image features, yielding a separate correlation map for each exemplar. To handle intra-image scale variations, the exemplar features are scaled to scales of 0.9 and 1.1, and the scaled exemplar features are correlated with the image features. All the correlation maps (2 conv blocks $\times$ 3 exemplars $\times$ 3 scales), including the one obtained at the original exemplar size, are concatenated, and passed along to the Density Prediction module. The correlation maps obtained from the conv4 block are smaller in size than the conv3 block, and we use bilinear interpolation to upsample them.

The Density Prediction module uses the correlation maps computed by the feature correlation module for predicting the density map F. The density prediction module consists of five convolutional blocks and three upsampling layers placed after the first, second, and third convolution layers. The last layer is a 1×1 convolution layer, which predicts the density map F. The size of F is the same as the size of the input image. We use LeakyReLU activation function and instance normalization after each of the intermediate convolution layer. LeakyReLU activation is used here so as to avoid *zero gradients* which could potentially affect the training of the generator, since the training signal for the generator comes from the counting network. We use ReLu activation after the last 1×1 convolution layer. Note that using the correlation maps as the input makes the density prediction module agnostic to the visual category and helps in generalizing to novel categories [29].

3.2. Training objective

A VCN is trained end-to-end by minimizing an objective function that is the combination of the three losses: counting loss, reconstruction loss, and diversity loss.

Counting loss. The counting loss is the Mean Squared Error (MSE) between the density map $f(X, B)$ predicted by regressor network and the Gaussian-smoothed image of the ground truth dot map Y. The ground truth dot map Y is a binary map where each object of interest is annotated with a dot. However, it is difficult to train counting networks directly on the 2D dot annotation map due to its sparsity and binary nature. Hence, most of the previous works first convolve the dot annotation map with a Gaussian kernel and train the counting network on the resulting density map. Since the size of the objects can vary a lot across the images of a few-shot counting dataset, we follow [29] and use an adaptive window size for the Gaussian kernel across different images. The size of the window is determined based on the average object size within an image, and we use the average distance between adjacent pairs of objects as a proxy for the average object size. Let $F = f(X, B)$ be the output density map and $\hat{Y}$ be the Gaussian-smoothed version of Y. The counting loss is formally defined as:

$$\mathcal{L}_{cnt}(X, B, Y) = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}(F(i,j) - \hat{Y}(i,j))^2. \quad (2)$$

However, using this MSE loss by itself does not force the generator to learn from the vicinity of the training samples. We will now describe two losses that force the generator to generate diverse samples from the vicinity of the training samples.

Reconstruction loss. This loss is designed to force the generator to generate samples which are *close* to the input training sample. For an image X, a noise vector z and a generated sample X_z, the reconstruction loss imposes a L_1

penalty between X_z and X.

$$\mathcal{L}_{rec}(X, X_z) = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}|X(i,j) - X_z(i,j)|. \quad (3)$$

where H, W are the height and width of the input image.

Diversity Loss. The generator may output the input X to minimize the reconstruction loss described above. However, the identity mapping from X to itself would imply zero augmentation, and this would not help the counting network in achieving better performance. To prevent the generator from replicating X, we pass two separate latent vectors z_1 and z_2 through the generator, resulting in samples X_{z_1} and X_{z_2} from the vicinity of X, and force the features of X_{z_1} and X_{z_2} to be different from one another. We pass X_{z_1} and X_{z_2} through the Feature Representation module, and obtain the convolutional features from the conv4 block of the Resnet-50 backbone. Let the feature maps be denoted by three-dimensional tensors $G_1 = g(X_{z_1})$ and $G_2 = g(X_{z_2})$. We subtract the corresponding mean vectors from the features, resulting in normalized features $\tilde{G}_1$ and $\tilde{G}_2$. Finally, we define the diversity loss as the dot product between the features in the two feature maps as

$$\mathcal{L}_{div}(X_{z_1}, X_{z_2}) = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}\langle\tilde{G}_1(i,j), \tilde{G}_2(i,j)\rangle. \quad (4)$$

$\tilde{G}_1(i,j)$ is the feature vector at spatial location i, j in the normalized feature $\tilde{G}_1$. The diversity loss $\mathcal{L}_{Diversity}$ loss imposes a large penalty if the two feature maps $g(X_{z_1})$ and $g(X_{z_1})$ are similar. Hence, the diversity loss encourages the generator to make the two samples X_{z_1} and X_{z_2} different from one another.

Overall training objective. The combined objective for training VCNs is:

$$\begin{aligned}&\lambda_1\mathcal{L}_{cnt}(X, B, Y) + \lambda_2\mathcal{L}_{cnt}(X_{z_1}, B, Y)\\ &+\lambda_3\mathcal{L}_{rec}(X, X_{z_1}) + \lambda_4\mathcal{L}_{div}(X_{z_1}, X_{z_2}),\end{aligned} \quad (5)$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are scalar weights associated with the different loss functions. We tune these scalar weights on the validation set of FSC-147 dataset [29].

3.3. Implementation details

Now we describe the details for reproducing our results. For training, we use Adam optimizer with a learning rate of 10^{-5}, batch size of 14, and random crops of size 384×384. The weights of the density prediction branch are initialized from a zero mean univariate Gaussian with standard deviation of 10^{-3}. The latent vectors z are 128 dimensional, and they are sampled from a univariate Gaussian with zero mean and unity variance. The number of hidden layer and output layer neurons in the mapping network h are 128 and 512 respectively. The values for $\lambda_1, \lambda_2, \lambda_3$ and λ_4 are 10^3, 10^2, 10^{-4}, 10^{-3} respectively. Note that the large values for λ_1, λ_2 does not mean that the other losses are ignored during the optimization. The ranges of the four losses are different, and these scalar weights scale these losses to make the average magnitude of the first loss term in the Eq. (5) to be an order of magnitude greater than the latter three loss terms. For training, we use three GPUs in parallel. The training is done for 1000 epochs.

4. Experiments

4.1. Few-shot counting experiments

4.1.1 Dataset and evaluation metrics

We perform experiments on the FSC147 dataset, which is the only dataset suitable for the few-shot counting task. This is a medium-scale dataset consisting of 6135 images from 147 visual categories. The dataset is divided into disjoint train, val, test sets with 3659, 1286, and 1190 images respectively. Note that there are no common categories between the train, val and test sets, so the counting methods would need to generalize to completely unseen classes at test time. Each image comes with three bounding box annotations for the exemplar, and dot annotations for all objects of interest in the image. The minimum, average, and maximum counts for the dataset are 7, 56, and 3701 respectively.

We use Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to measure the accuracy of a counting method. MAE and RMSE are commonly used metrics for counting task [25, 27, 47], and they are defined as follows. $MAE = \frac{1}{n}\sum_{i=1}^{n}|c_i - \hat{c}_i|; RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(c_i - \hat{c}_i)^2}$, where n is the number of test images, and c_i and $\hat{c}_i$ are the ground truth and predicted counts.

4.1.2 Comparison with few-shot detection and counting approaches

We compare VCN with several state-of-the-art few-shot detection and counting approaches on the Val and Test splits of FSC147. The few-shot object detectors [8, 14] follow the detect then count approach rather than our density map prediction approach. The few-shot counting methods [9,24,29] follow a density map prediction approach, similar to ours. Table 1 shows the comparison results. VCN is our proposed approach, where we discard the generator and pass the test image directly through the regressor. VCN* employs the test time adaptation strategy proposed in [29], where the exemplars from the test image are used to adapt the regressor to the novel test class. Note that FamNet also follows the same test time adaptation strategy. As can be seen from the

Method	Val Set		Test Set	
	MAE	RMSE	MAE	RMSE
FR few-shot detector [14]	45.45	112.53	41.64	141.04
FSOD few-shot detector [8]	36.36	115.00	32.53	140.65
GMN [24]	29.66	89.81	26.52	124.57
MAML [9]	25.54	79.44	24.90	112.68
FamNet [29]	23.75	69.07	22.08	99.54
VCN (Proposed)	20.24	62.99	19.30	99.08
VCN* (Proposed)	**19.38**	**60.15**	**18.17**	**95.60**

Table 1. **Comparing VCN to previous state-of-the-art few-shot approaches on FSC147 dataset**, including few-shot detectors [8, 14] and few-shot counting methods [9, 24, 29]. Note that both FamNet and VCN* use the test time adaptation strategy proposed by [29]. VCN does not use any test time adaptation. Both VCN and VCN* outperform all of the approaches on both Val and Test splits.

table, both VCN and VCN* outperform all of the previous approaches on both the Val and Test splits. Furthermore, the few-shot detection based methods [8, 14] perform worse than the density map based approaches. VCN and VCN* outperform FamNet [29] by a large margin, even though the architecture of FamNet is same as that of our few-shot regressor. This performance gain suggests the usefulness of our proposed data augmentation strategy.

4.1.3 Comparison with pre-trained object detectors

Object detectors can be used for visual counting by simply counting the number of detections in an image. However, object detectors can only be used for a limited set of categories on which they are trained. This makes it harder for object detectors to count a large number of visual categories. Furthermore, even for the categories that the object detectors are trained on, the results of the detectors might still be far from perfect, and this lead to poor counting results. We compare VCN with several object detectors on FSC147-Val-COCO and FSC147-Test-COCO sets, which are subsets of images from FSC147 Val and Test set which share categories with MSCOCO dataset [21]. The results are reported in Table 2. We compare VCN with object detectors [11, 20, 32] trained on MSCOCO dataset and also the state-of-the-art fewshot counting network FamNet [29]. The detectors are implemented in the Detectron2 [43] library. Even though the detectors are trained on a large number of training images from the MSCOCO dataset, VCN outperforms the detectors on both of the data splits, as can be seen in Table 2.

4.1.4 Comparison with competing augmentation strategies

In this section, we compare VCN with six data augmentation strategies: simple augmentation strategy of adding Gaussian noise to the input image, random colour jittering, PCA lighting variation, AutoAugment [7], Mixup [46], and Manifold Mixup [37]. AutoAugment uses a Reinforcement Learning based search over a large search space consisting of around 10^{34} augmentation policies for learning an optimal data augmentation policy, and requires thousands of GPU hours for the search. We use the publicly available Imagenet policy, since the Imagenet policy was shown to generalize well to other tasks and datasets [7]. Other than the style transfer perspective described earlier, another way of looking at the VCN generator is that it combines a random noise vector with an image. The first competing approach of Gaussian noise addition is to check if simply adding noise to the input image can lead to performance gains. Mixup and Manifold Mixup are popular augmentation strategies which have been shown to useful for tasks such as image classification.

The results are presented in Table 3. Addition of Gaussian Noise performs the worst, which shows that simply adding noise to an input image does not capture well the vicinity distribution around training samples. We also see that Mixup, Manifold Mixup and AutoAugment perform worse than No Augmentation. This shows that the these three strategies which have shown significant improvements on image classification task may not be suitable for the pixel level prediction task of few-shot counting.

4.1.5 Ablation study on the training objective

Recall that the overall training objective of VCN consists of count loss, reconstruction loss and diversity loss. In Table 4, we analyze the usefulness of each of these loss terms. As can be seen from the table, adding each of the loss term leads to improvement in performance. Please see the Supplementary material for an ablation study on the number of exemplars.

Method	FSC147-Val-COCO		FSC147-Test-COCO	
	MAE	RMSE	MAE	RMSE
Faster R-CNN	52.79	172.46	36.20	79.59
RetinaNet	63.57	174.36	52.67	85.86
Mask R-CNN	52.51	172.21	35.56	80.00
FamNet	39.82	108.13	22.76	45.92
VCN (proposed)	**34.15**	**97.78**	**21.81**	**45.85**

Table 2. **Comparing Vicinal Counting Nets with pre-trained object detectors**, on FSC147-Val-COCO and FSC147-Test-COCO splits of FSC147 which contain images from COCO classes. VCN outperforms all the object detectors and FamNet [29] on the COCO classes.

Augmentation strategy	MAE	RMSE
No Augmentation	21.14	69.88
Gaussian Noise	24.65	79.91
Colour Jittering	21.18	70.26
PCA lighting variation	20.95	66.58
Auto Augment [7]	21.37	78.43
Mixup [46]	22.28	78.38
Manifold Mixup [37]	21.66	73.92
VCN (Proposed)	**20.24**	**62.99**

Table 3. **Comparing different augmentation strategies on the Val set of FSC147**. For fair comparison, we do not use test time adaptation for any of the methods. VCN outperforms the other strategies on both MAE and RMSE metrics.

Loss	Combinations		
Count Loss	✓	✓	✓
Reconstruction Loss	✗	✓	✓
Diversity Loss	✗	✗	✓
MAE	21.14	20.51	20.24
RMSE	69.88	66.07	62.99

Table 4. **Analyzing the training losses on the Val Set of FSC147**. Each of the loss terms lead to improved results.

4.1.6 Qualitative results

In Fig. 3, few images and the corresponding generated images are shown. The generator leaves the location of the objects unchanged, and changes the color of the object and the background. Some of the success and failure cases of VCN are shown in Fig. 4.

4.2. Crowd-counting experiments

Although the main focus of our work is on few-shot counting, VCN can prove to be useful for other pixel level prediction tasks like crowd counting. We conduct experiments on the Shanghitech Part A [47], UCF QNRF [13] and NWPU [41] datasets, which are popular crowd counting datasets. Shanghaitech Part A, UCF QNRF and NWPU

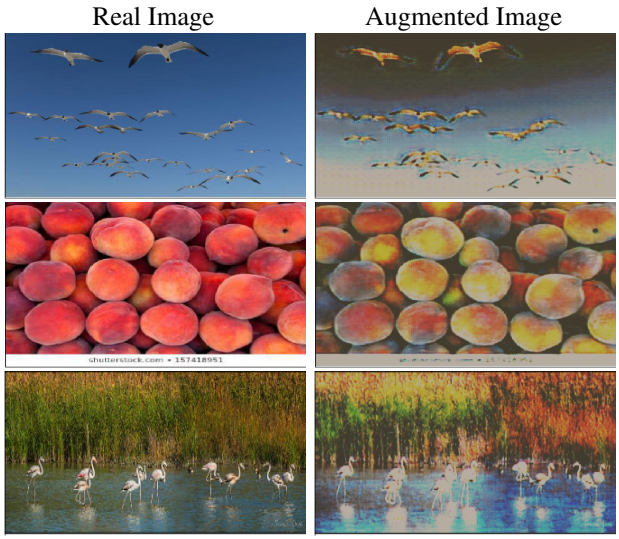

Figure 3. **Few images and augmented pairs.**

datasets consist of 482, 1535 and 5109 images respectively. Since the regressor network of VCNs are designed for few shot counting task, we replace the regressor with that of DM-Count [40] and simply add the reconstruction and diversity loss on top all of the counting losses used in DM-Count. We report the results in Table 5. On Shanghaitech Part A and NWPU datasets, VCN outperforms all of the previous methods in terms of both the MAE and RMSE metrics. On UCF QNRF, VCN outperforms all of the methods in terms of MAE metric. This shows the usefulness of our data augentation strategy for tasks other than few-shot counting.

5. Discussion

In this paper, we considered the task of few-shot counting, and we tackled the issue of limited training set size of the existing few-shot counting datasets. We presented Vicinal Counting Network, a novel architecture for jointly learning to count and augment the training samples. Using the proposed architecture, we advanced the state-of-the-art performance for few-shot counting. We also demonstrated the

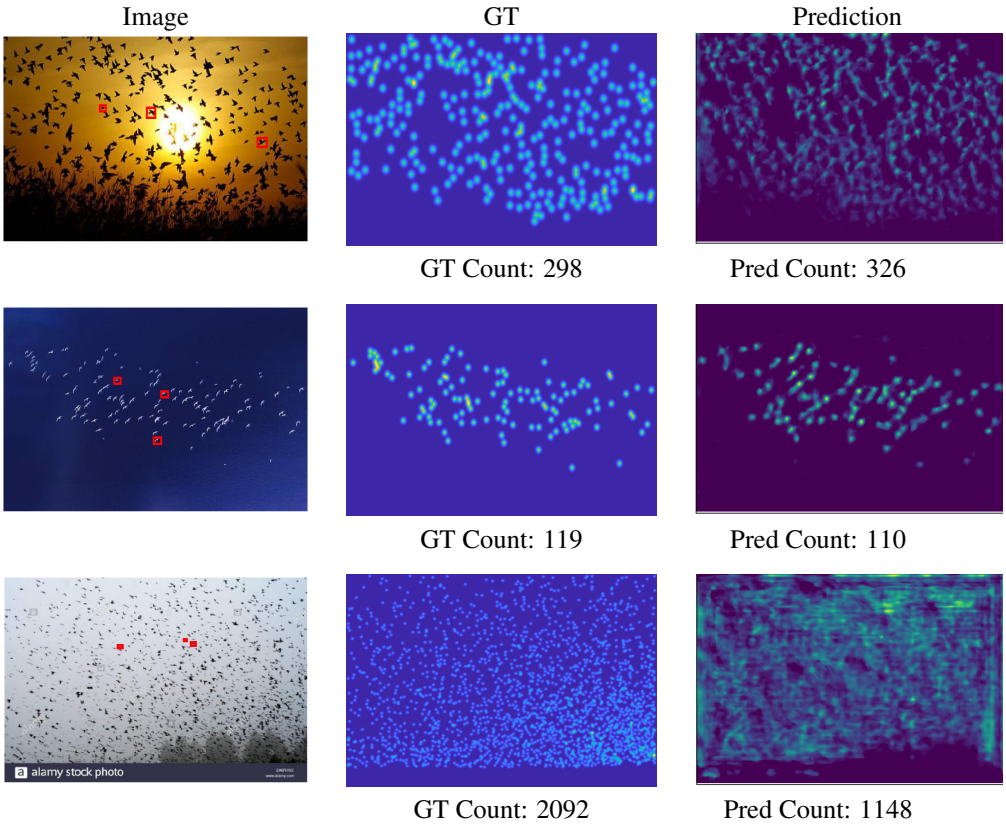

Figure 4. **Some representative success and failure cases of VCN**. VCN performs well on first two test cases, but does not perform well on the last test case.

	Shanghaitech Part A		UCF QNRF		NWPU Val	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
MCNN [47]	110.2	173.2	277	426	218.5	700.6
Switching CNN [33]	90.4	135.0	228	445	-	-
CP-CNN [36]	73.6	106.4	132	191	-	-
IG-CNN [4]	72.5	118.2	-	-	-	-
ic-CNN [27]	68.5	116.2	-	-	-	-
SANet [5]	67.0	104.5	-	-	-	-
CSR Net [19]	68.2	115.0	-	-	104.8	433.4
CAN [22]	62.3	100.0	107	183	93.5	489.9
SFCN [42]	64.8	107.5	102	171	95.4	608.3
ANF [45]	63.9	99.4	110	174	-	-
Bayesian Loss [25]	62.8	101.8	89	155	93	470
TopoCount [1]	61.2	104.6	89	159	-	-
DM-Count [40]	59.7	95.7	85.6	**148.3**	70.5	357.6
VCN (Proposed)	**57.1**	**95.4**	**85.2**	148.8	**62.0**	**207.7**

Table 5. **Count errors of different methods on the Shanghaitech Part A, UCF QNRF and the validation set of NWPU datasets.** We report both MAE and RMSE metrics. VCN augmentation strategy leads helps in achieving better crowd counting performance.

applicability and benefits of our method to crowd counting task. However, when compared to our improvement gains on the Few-Shot Counting task, the improvements from our method on the crowd counting task were somewhat less.

References

[1] Shahira Abousamra, Minh Hoai, Dimitris Samaras, and Chao Chen. Localization in the crowd with topological constraints. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2021. 2, 8

[2] Carlos Arteta, Victor Lempitsky, J Alison Noble, and Andrew Zisserman. Detecting overlapping instances in microscopy images using extremal region trees. *Medical image analysis*, 27:3–16, 2016. 2

[3] Carlos Arteta, Victor Lempitsky, and Andrew Zisserman. Counting in the wild. In *Proceedings of the European Conference on Computer Vision*, 2016. 2

[4] Deepak Babu Sam, Neeraj N Sajjan, R Venkatesh Babu, and Mukundhan Srinivasan. Divide and grow: Capturing huge diversity in crowd images with incrementally growing cnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2, 8

[5] Xinkun Cao, Zhipeng Wang, Yanyun Zhao, and Fei Su. Scale aggregation network for accurate and efficient crowd counting. In *Proceedings of the European Conference on Computer Vision*, 2018. 2, 8

[6] Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. Vicinal risk minimization. 2001. 1, 3

[7] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019. 6, 7

[8] Qi Fan, Wei Zhuo, Chi-Keung Tang, and Yu-Wing Tai. Few-shot object detection with attention-rpn and multi-relation detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2, 5, 6

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning*, 2017. 2, 3, 5, 6

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*. 2014. 2

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the International Conference on Computer Vision*, 2017. 6

[12] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the International Conference on Computer Vision*, 2017. 3

[13] Haroon Idrees, Muhmmad Tayyab, Kishan Athrey, Dong Zhang, Somaya Al-Maadeed, Nasir Rajpoot, and Mubarak Shah. Composition loss for counting, density map estimation and localization in dense crowds. In *Proceedings of the European Conference on Computer Vision*, 2018. 7

[14] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *Proceedings of the International Conference on Computer Vision*, 2019. 2, 5, 6

[15] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 3

[16] Aisha Khan, Stephen Gould, and Mathieu Salzmann. Deep convolutional neural networks for human embryonic cell counting. In *Proceedings of the European Conference on Computer Vision*, 2016. 2

[17] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, 2015. 2

[18] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. 2

[19] Yuhong Li, Xiaofan Zhang, and Deming Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1, 2, 8

[20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the International Conference on Computer Vision*, 2017. 6

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision*, 2014. 6

[22] Weizhe Liu, Mathieu Salzmann, and Pascal Fua. Context-aware crowd counting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2, 8

[23] Xialei Liu, Joost Van De Weijer, and Andrew D Bagdanov. Leveraging unlabeled data for crowd counting by learning to rank. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[24] Erika Lu, Weidi Xie, and Andrew Zisserman. Class-agnostic counting. In *Proceedings of the Asian Conference on Computer Vision*, 2018. 2, 5, 6

[25] Zhiheng Ma, Xing Wei, Xiaopeng Hong, and Yihong Gong. Bayesian loss for crowd count estimation with point supervision. In *Proceedings of the International Conference on Computer Vision*, 2019. 1, 2, 5, 8

[26] T Nathan Mundhenk, Goran Konjevod, Wesam A Sakla, and Kofi Boakye. A large contextual dataset for classification, detection and counting of cars with deep learning. In *Proceedings of the European Conference on Computer Vision*, 2016. 2

[27] Viresh Ranjan, Hieu Le, and Minh Hoai. Iterative crowd counting. In *Proceedings of the European Conference on Computer Vision*, 2018. 1, 2, 5, 8

[28] Viresh Ranjan, Mubarak Shah, and Minh Hoai Nguyen. Crowd transformer network. *arXiv preprint arXiv:1904.02774*, 2019. 1

[29] Viresh Ranjan, Udbhav Sharma, Thu Nguyen, and Minh Hoai. Learning to count everything. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 3, 4, 5, 6, 7

[30] Viresh Ranjan, Boyu Wang, Mubarak Shah, and Minh Hoai. Uncertainty estimation and sample selection for crowd counting. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 1

[31] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016. 2

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015. 6

[33] Deepak Babu Sam, Shiv Surya, and R Venkatesh Babu. Switching convolutional neural network for crowd counting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 8

[34] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. One-shot learning with memory-augmented neural networks. *arXiv:1605.06065*, 2016. 2

[35] Miaojing Shi, Zhaohui Yang, Chao Xu, and Qijun Chen. Revisiting perspective information for efficient crowd counting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2

[36] Vishwanath A Sindagi and Vishal M Patel. Generating high-quality crowd density maps using contextual pyramid cnns. In *Proceedings of the International Conference on Computer Vision*, 2017. 8

[37] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *Proceedings of the International Conference on Machine Learning*, 2019. 1, 3, 6, 7

[38] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016. 2

[39] Jia Wan and Antoni Chan. Adaptive density map generation for crowd counting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1130–1139, 2019. 2

[40] Boyu Wang, Huidong Liu, Dimitris Samaras, and Minh Hoai. Distribution matching for crowd counting. In *Advances in Neural Information Processing Systems*. 2020. 1, 7, 8

[41] Qi Wang, Junyu Gao, Wei Lin, and Xuelong Li. Nwpu-crowd: A large-scale benchmark for crowd counting. *arXiv preprint arXiv:2001.03360*, 2020. 7

[42] Qi Wang, Junyu Gao, Wei Lin, and Yuan Yuan. Learning from synthetic data for crowd counting in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2, 8

[43] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2, 2019. 6

[44] Weidi Xie, J Alison Noble, and Andrew Zisserman. Microscopy cell counting and detection with fully convolutional regression networks. *Computer methods in biomechanics and biomedical engineering: Imaging & Visualization*, 6(3):283–292, 2018. 2

[45] Anran Zhang, Lei Yue, Jiayi Shen, Fan Zhu, Xiantong Zhen, Xianbin Cao, and Ling Shao. Attentional neural fields for crowd counting. In *Proceedings of the International Conference on Computer Vision*, 2019. 2, 8

[46] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 1, 3, 6, 7

[47] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 2, 5, 7, 8