Image-to-Image Translation with Conditional Adversarial Networks

Pengkai Zhu zpk@bu.edu
Weiwei Tao wwtao@bu.edu
Hari Saran chhari@bu.edu

Abstract

We present a conditional generative adversarial network to solve image-to-image translation problems. The generator utilizes the U-Net model - an encoder-decoder network with skips to shuttle low-level information between layers. We combine the pixel-wise reconstruction distance with the adversarial loss as the objective function to train the framework.We not only qualitatively but also quantitatively demonstrate the effectiveness of our model on synthesizing photos from labels, maps and reconstructing objects from edge maps.

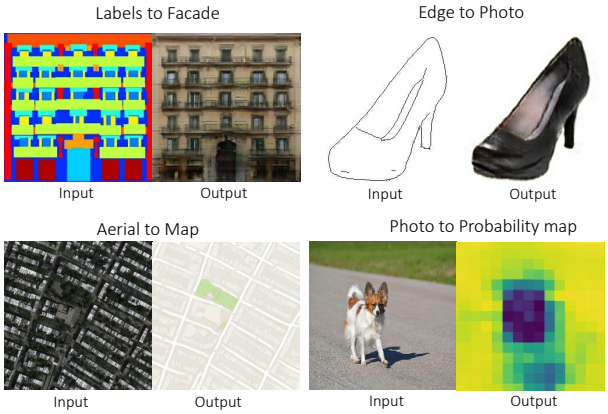

Figure 1: Image-to-image translation, defined as translating one representation of a scene into another, includes may problems in image processing, such as inpainting, style transfer and image prediction from a normal map. Even though these problems are always mapping pixels to pixels, they are often tackled by different application-specific algorithms. We build a conditional adversarial net as a general-purpose solution to deal with all these image translation problems. Selected results generated by our cGAN model are shown here.

1. Task

Many problems in image processing and computer vision, such as inpainting, style transfer and image prediction from a normal map, can be considered as translating an input image to a corresponding output image. In this project, we will implement a general framework, conditional generative adversarial networks (cGAN), to address these problems. Different from other application-specific solutions, this framework only needs a high-level objective and is able to learn the appropriate loss function automatically. We apply this model to several datasets with different tasks, and both the qualitative and quantitative evaluation results demonstrate the effectiveness of this method.

2. Related Work

Many researches [9, 17, 5] formulate image-to-image translation problems as per-pixel classification or regression. However, these formulations cannot capture the structured information between pixels. Structured losses, which penalize the joint configuration of the output, are combined with many popular methods as reported in [1], [3] and [7].

The convolutional neural networks (CNNs) architecture is a popular tool used in image-to-image translation problems [10]. Zhang et al. [19] utilized CNN to generate colorful images given a grayscale photograph as input. They treat the colorization problem as a classification task and the distribution of possible colors for each pixel was predicted. The loss was re-weighted during training so as to increase the diversity of colors in the results. Their model has shown the ability to produce results indistinguishable from real color photos by training over one million color images. However, the performance of CNN is strongly dependent on loss functions specified by users. Common-used loss function for CNN, such as the Euclidean distance will tend to produce conservative and blurry images [10].

Since the goal is to make the output indistinguishable from reality, the Generative Adversarial Networks (GANs) becomes popular approach in image prediction problems. GAN, proposed by Goodfellow et al. [4], introduces a generator, which generates images and a discriminator, which distinguishes between generated samples and real images. The competition between the generators and the discriminators drives the networks to be optimized until the generated

images are indistinguishable from the ground truth. GANs are able to improve image quality since blurry images look obviously fake. Nevertheless, training GANs is known to be unstable, often resulting in generators that produce incomprehensible images.

Extensions of GANs were developed for image-to-image translation problems. Radford et al. [11] recently proposed architectures named Deep Convolutional GANs (DCGANs), which is relatively more stable to be trained and have succeeded in generating high quality images. Denton et al. [2] developed the conditional versions of the GAN model. They use a cascade of GANs to generate images in a coarse-to-fine fashion. Their approach is proved to be capable to produce high quality samples of natural images for large datasets. Xiaolong et al. [2] learned a model by first training two convolutional GANs where the image generation process is factorized into the generation of 3D structure and style. The conditional GAN (cGAN) haven been widely applied in various image prediction problems such as image inpainting [10], image prediction from a normal map [16], product photo generation [18] and style transfer [8].

3. Approach

GANs are generative models that work via the principle of maximum likelihood where the probability that the model assigns to the training data is maximized. Instead of explicitly defining a density function, GANs offer a method to train the model while sampling data from the probability distribution. A GAN consists of two 'adversarial' models: a generative model G to synthesize outputs resembling real images and a discriminative model D to distinguish real images from synthesized ones.

3.1. Loss function

Our approach is to train a conditional GAN model where both the generator and discriminator are conditioned on the input image. As illustrated in Fig. 2, the learning framework is a two-player game where the discriminator D tries to distinguish real training data from synthetic images and the generator G tries to confuse D by producing samples that appear as "real" as possible. Concretely, D and G can be trained jointly by solving

$$\mathcal{L}_{cGAN}(G,D) = E_{x,y\sim p_{data}(x,y)}[\log D(x,y)] + E_{x\sim p_{data}(x),z\sim p_z(z)}[\log(1-D(x,G(x,z)))] \quad (1)$$

Previous study has proposed that mix the Adversarial loss with a traditional L_2 loss gives better prediction in image inpainting task[10]. Thus, we combine $\mathcal{L}_{cGAN}(G,D)$ with the Euclidean distance as our objective function

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_1(G)$$

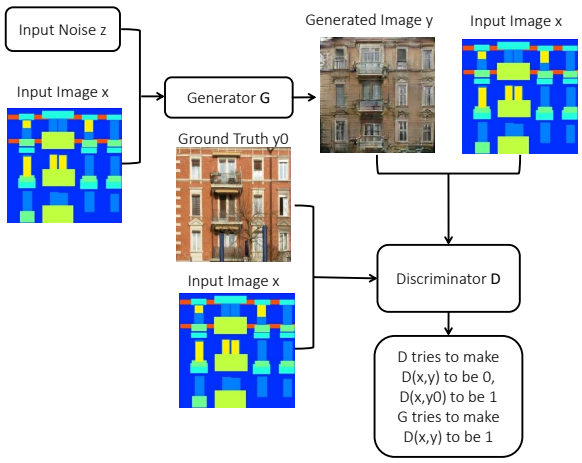

Figure 2: The cGAN framework to predict architectural photos from labels. The discriminator is trained to distinguish between real and synthesized images and the generator is trained to fool the discriminator. Both the generator and discriminator are conditioned on an input image.

where L_1 distance is utilized since L_1 encourages less blurring than L_2:

$$\mathcal{L}_1(G) = E_{x,y\sim p_{data}(x,y),z\sim p_z(z)}[||y - G(x,z)||_1] \quad (2)$$

3.2. Encoder-decoder with skips

The structure of generator we used is a variation of encoder-decoder network. Given an input image of size 256×256 with 3 color channels, we use a series of encoders, which consist of a stride-2 convolutional layer, spatial batch normalization and a rectified linear unit (ReLU) activation, to perform downsampling until a bottleneck layer.The original input feature is reconstructed to a 1×1 hidden layer with 512 channels.

In our image translation problem, the input and output are different in their appearance, but share the same underlying structure. To improve the accuracy of image-to-image translation, the low-level infromation, such as the location of prominent edges in an image colorization problem, should be shuttled across the net. However, because of downsampling, feature information is circumvented by the bottleneck to pass through all the layer. To alleviate this issue, we add skip connections between each layer i and layer $n-i$, where n is the total number of layers k as illustrated in Fig. 3[12].

The second half of the pipeline is the decoders, which generate pixels of the output image using encoder features. Each decoder consists of an up-convolution that halves the

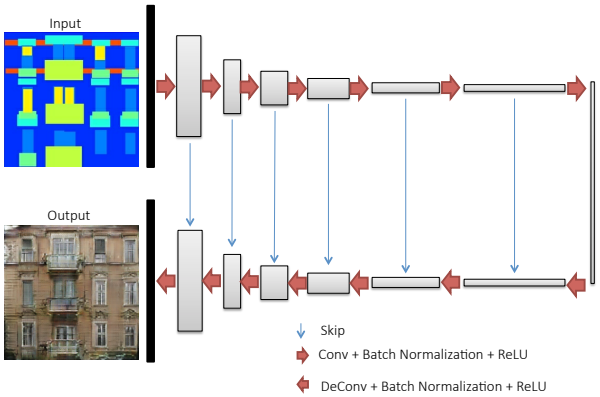

Figure 3: U-net architecture. Each gray box corresponds to a multi-channel feature map. Right-oriented arrows denote the encoder while the left-oriented arrows denote the decoder. The blue arrows represent skip connections between mirrored layers in the encoder and decoder stacks.

number of feature channels, a concatenation with corresponding copied feature map from the skip connection and a batch normalization, followed by a ReLU except for the last layer, where TanH activation is applied. Each skip connection concatenates all channels at layer i with those at layer n-1.

Layers in discriminator also utilized the convolution-BatchNorm-ReLu modules except for the last layer, which is a convolution with one output channel followed by the sigmoid activation. While the input is the concatenation of the input image and an unknown image, the output of the discriminator is a 30×30 image where each pixel value represents how believable the corresponding section of the unknown image is.

4. Experimental Results

We followed Isola et al. [6] to perform this final project.

4.1. Dataset

Since our goal is to build a general framework which can solve all kinds of image-to-image translation problems, we test the model on a variety of datasets for different tasks:

- Labels↔architectural photo: CMP Facades dataset [15].
- Map↔ aerial photo: data scraped from Google Maps.
- Edges→shoes photo: data from [20] for edge to shoes.
- Photo→probability map: ImageNet validation dataset [13].

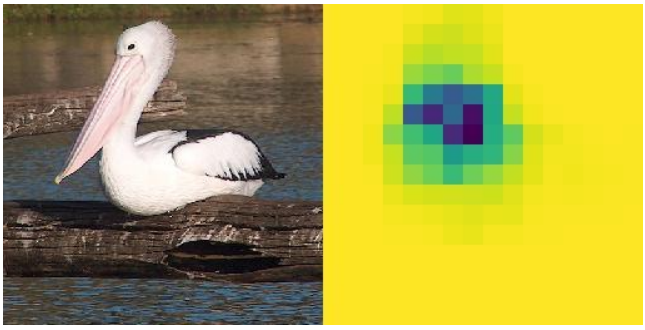

Figure 4: Example of an image and its probability map.

The first three datasets are obtained from [6]. Mapping images to probability maps is originally designed for classification tasks. In a classification problem, only part of the image is helpful to the classification. For instance, in Fig. 4 (a), the pelican stays in the center of the image, the rest is background which is useless for classification. Taking a small square to block part of the the image, change the block position, the probability of true label of the occluded image is shown in Fig. 4 (b). Yellow represents high probability and blue represents low, which proves that the loss of the background does not affect the classification accuracy.

If we can classify images by analyzing part of the image, the efficiency of classification would be substantially improved. Therefore, we designed a task use the cGAN to generate probability maps from images in the ImageNet validation dataset. The dataset contains 500 categories. Each category has 20 photo+probability image pairs in the training set and 5 image pairs in the validation set. The exact number of training samples and validation samples is summarized in Table 1.

4.2. Implementation details

The pipeline was implemented in Tensorflow. In Facades and GoogleMap dataset, we trained the model in both from labels/maps to photo and from photo to labels/maps. For mapping from edges to shoes photo and from image to probability map, the model is trained in one direction.

Table 2 illustrates the generator and discriminator networks. Given a $256 \times 256 \times 3$ input image, we perform 8 layers of convolutional operations and generate a bottleneck layer with size of $1 \times 1 \times 512$. The bottle neck layer then passes through 8 layers of deconvolutional operations. The output image size is restricted to $256 \times 256 \times 3$.The discriminator, a 5-layer network, takes the concatenation of an input image and generated/target image as input and outputs a $30 \times 30 \times 1$ feature map which predicts the image is real or generated.

Table 1: Number of training samples and validation samples for each dataset used in our simulation.

Dataset	Training Samples	Validation Samples
Facades	400	100
Google Maps	1,096	1,098
Edge to Shoes	16k	200
Photo to probability map	10,000	2,500

Table 2: Network architectures. Top: generator of cGAN; bottom: discriminator of cGAN (left). conv means convolutional layer, uconv means fractionally-strided convolutional (deconvolutional) layer.

Generator Encoders	conv1	conv2	conv3	conv4	conv5	conv6	conv7	conv8
Input size	256×256	128×128	64×64	32×32	16×16	8×8	4×4	2×2
Input channel	3	64	128	256	512	512	512	512
Generator Decoders	unconv1	unconv2	unconv3	unconv4	unconv5	unconv6	unconv7	unconv8
Input size	1×1	2×2	4×4	8×8	16×16	32×32	64×64	128×128
Input channel	512	512	512	512	512	256	128	64

Discriminator	conv1	conv2	conv3	conv4	conv5
Input size	256×256	128×128	64×64	32×32	31×31
Input channel	6	64	128	256	512

We used the minibatch stochastic gradient descent solver, ADAM for optimization. The learning rate is set to 0.0002. We use a batch size of 1 and train for 200 epochs for Facades, GoogleMap. Because of time restrictions, the edges to shoes photo and Imagenet dataset is trained for 20 epochs. The simulation is implemented on a GTX TITAN X graphic card.

4.3. Evaluation metrics

The goal for image-to-image translation is to generate reasonably realistic images. How to evaluate the quality of synthetic images is still an open and difficult problem [14]. In order to evaluate the visual quality of our results, we employ two metrics.

4.3.1 Qualitative results

The first metric we used is the plausibility to people. For tasks like mapping labels to architectural photo and edges to shoes photo, the plausibility to a human is the ultimate goal. Figures 6, 7, 8 show selected results. As we can see, the cGAN can generate realistic images which match well with the ground truth images. We show that our model can generate a different appearance of image which share the same underlying structure with the input image.

However, if an input image is twoo complicated, the results get worse. We show some not very successful examples in 10 and 11. For figures in 10, cGAN can't capture the details in the input images. This problem may be caused by limited number of training data and training epochs. For 11, the cGAN did not capture the core region in an input image. The reason for the failure may be that there are multiple objects in the input image or the features for correct classification of the object is different with its main features.

4.3.2 Mean absolute error

To quantify the performance of the cGAN model, we calculate the pixel by pixel mean absolute error (MAE) between the lightness of the generated image and that of the ground truth corresponding to all input images in each dataset. The MAE is then averaged across all the images to get a mean value for each dataset. The estimation error for each dataset is summarized in Table 3. Obviously, for images with simpler structures, such as edges to shoes photo, have a smaller MAE comparing to more complicated tasks. Thus, MAE is a reasonable metric to evaluate the generated images by our cGAN model.

4.4. Loss function analysis

As we mentioned in Section 3, the loss function in our model includes two parts: the reconstruction L_1 distance and the adversarial part. We run studies to test the effect of L_1 term by setting λ to 0 and changing L_1 distance to L_2 distance, on the Facade dataset in the label to photo direction.

Fig. 5 shows the effects of these variations on mapping architectural labels to photo task. cGAN with L_2 results in some artifacts in facade synthesize, but the results are still comparable to cGAN with L_1. However, cGAN alone gives very blurry results. We also calculate the MAE for different cases. The MAE for cGAN+L_2 is 0.1778 ± 0.0405 which is

Table 3: MAE between the lightness of the generated image and the ground truth for each dataset.

Dataset	MAE
Labels to architectural photo	0.1757 ± 0.0420
Architectural photo to labels	0.1006 ± 0.0278
Aerial photo to map	0.0262 ± 0.0086
Map to aerial photo	0.1306 ± 0.0250
Edges to shoes photo	0.0765 ± 0.0250
Photo to probability map	0.2229 ± 0.0831

Figure 5: Results predicted by models trained with different losses on Facade dataset.

comparable to that for cGAN+L_1 as 0.1757$\pm$0.0420. However, the results with cGAN only have a much larger mean absolute error of 0.1930 and higher uncertainty of 0.0525. We can conclude that combining the cGAN loss with the reconstruction distance gives more realistic results.

5. Conclusion

In this work we develop a conditional adversarial network for image translation problems. We demonstrate that this model can synthesize plausible images for many image-to-image translation tasks. We compare the results with different losses and validate that a combination of the cGAN loss with the reconstruction distance gives more realistic results.

6. Detailed Roles

The detailed roles are summarized in github link.

7. Code Repository

The Github link for our code is:
https://github.com/chhari/ImagetoImageTranslation.git

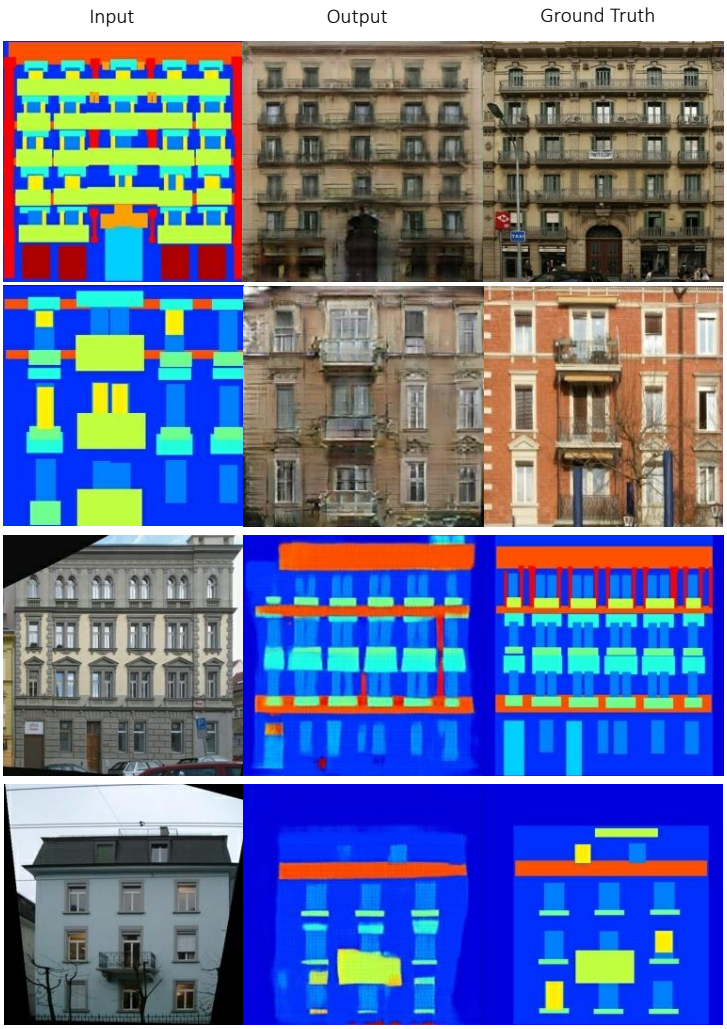

Figure 6: Examples of synthetic images on labels↔architectural photo, compared to ground truth.

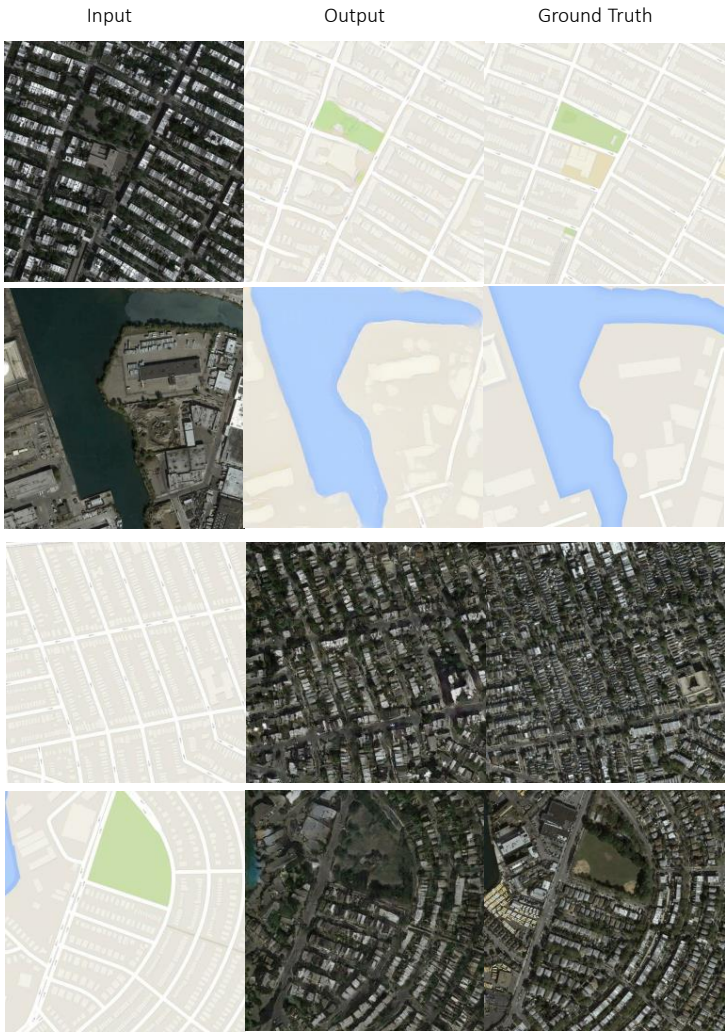

Figure 7: Example results of synthetic images on map↔ aerial photo, compared to ground truth.

Figure 8: Example results of synthetic images on edges→shoes photo, compared to ground truth.

Figure 9: Example results of synthetic images on photo→probability map, compared to ground truth.

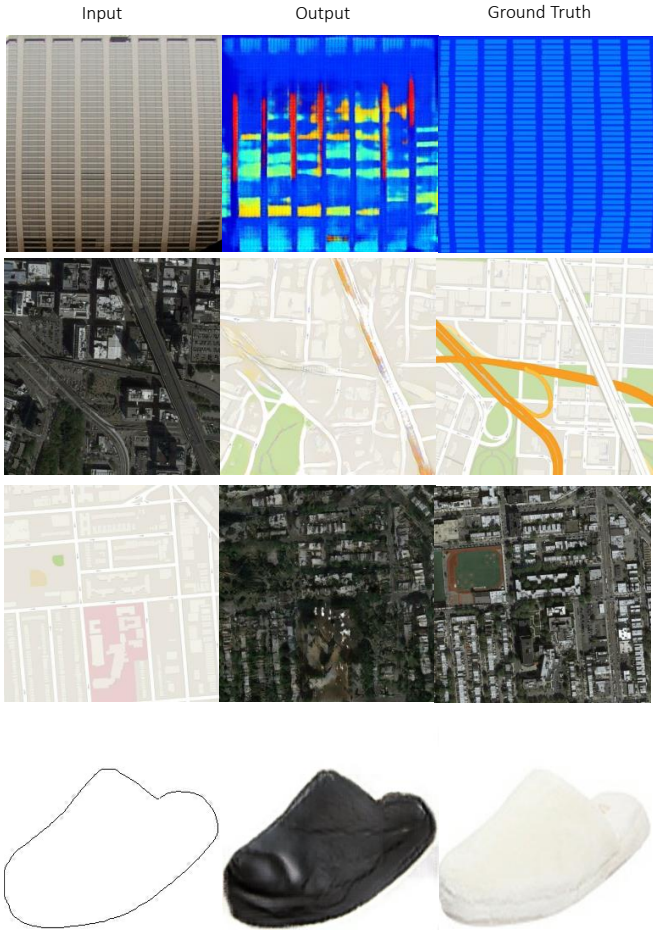

Figure 10: Example failure cases on labels↔architectural photo, map↔ aerial photo, edges→shoes photo. Common failures include difficulty in capturing the details in inputs images and twisting lines in input figures.

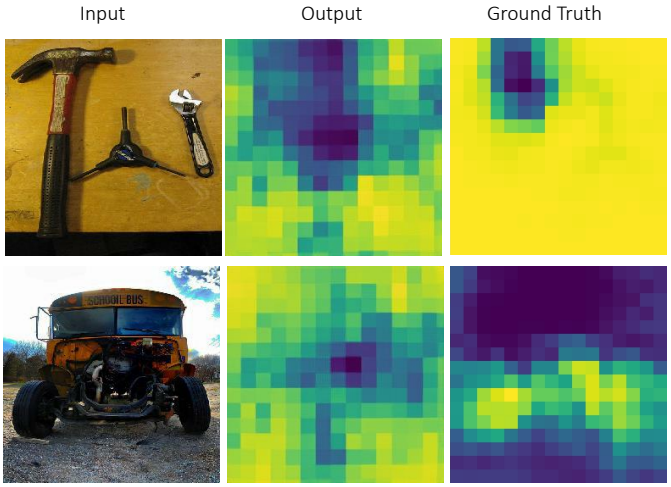

Figure 11: Example failure cases on photo→probability map.