我们提出eConv和ePool层，预测dense和相等的cnn模型。

Dense CNN Learning with Equivalent Mappings

Jianxin Wu **Chen-Wei Xie** **Jian-Hao Luo**
National Key Laboratory for Novel Software Technology, Nanjing University
163 Xianlin Avenue, Qixia District, Nanjing 210023, China
{wujx,xiecw,luojh}@lambda.nju.edu.cn

Abstract

Large receptive field and dense prediction are both important for achieving high accuracy in pixel labeling tasks such as semantic segmentation. These two properties, however, contradict with each other. A pooling layer (with stride 2) quadruples the receptive field size but reduces the number of predictions to 25%. Some existing methods lead to dense predictions using computations that are not equivalent to the original model. In this paper, we propose the equivalent convolution (eConv) and equivalent pooling (ePool) layers, leading to predictions that are both dense and equivalent to the baseline CNN model. Dense prediction models learned using eConv and ePool can transfer the baseline CNN's parameters as a starting point, and can inverse transfer the learned parameters in a dense model back to the original one, which has both fast testing speed and high accuracy. The proposed eConv and ePool layers have achieved higher accuracy than baseline CNN in various tasks, including semantic segmentation, object localization, image categorization and apparent age estimation, not only in those tasks requiring dense pixel labeling.

1 Introduction

Large receptive filed and densely repeated nonlinear mapping (dense predictions) are both important properties of many successful deep convolutional neural network (CNN) models. These two properties are, however, not compatible with each other in classic CNN models.

For example, in the VGG-16 CNN model [18], the output after the `pool5` layer is a $7 \times 7 \times 512$ tensor, each element in this order 3 tensor is computed from a 212×212 receptive field in the input image (which is 224×224 in size). In CNN models for semantic segmentation, the last layer's receptive field is even larger. For example, every output of the fully convolutional network (FCN) [12] is computed from a 404×404 receptive field. A larger receptive field incorporates more information from the input image and often leads to more accurate recognition or segmentation results. The increase of receptive field size is mainly achieved by the pooling layers (with stride > 1) in current CNN models. A 2×2 pooling layer with stride 2 will quadruple the receptive field size, and a 3×3 convolution with stride 1 will increase the receptive field size by 2 both horizontally and vertically.

When the input image size is larger than one layer's receptive field size, the same nonlinear mapping is applied to different regions in the input images. For example, suppose for a large input image, the `pool5` layer output $(i, j, :)$ is computed by a nonlinear mapping f using a 212×212 image window whose top left corner is (x, y) . Then, the `pool5` layer output $(i + 1, j, :)$ is computed by the *same* nonlinear mapping f using the image window at $(x + 32, y)$ (if padding is used, the exact receptive field might change). Applying the same mapping many times to obtain dense predictions is important to improve CNN accuracies too. For example, CNN models for semantic segmentation apply the same nonlinear mapping to different regions of an input image to obtain pixel labeling results.

Large receptive field size and dense prediction, however, contradict with each other. One pooling layer will in general quadruple the receptive filed size, but reduces the number of predictions to a

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

quarter of the number before the pooling layer. In the FCN model, the number of nonlinear mapping outputs (predictions) is reduced to only 16×16 for a 500×500 input image (700×700 after padding), which has to be upsampled to roughly 32 times to compare with the 500×500 groundtruth. In other words, approximately every 1000 pixels is given only one prediction by the FCN model before the upsampling. The upsampling step is an interpolation, which does not increase the real number of predictions.

Many methods have been proposed to make the nonlinear mapping f be computed more times, i.e., to make *denser* predictions, mainly in object detection and semantic segmentation tasks. These methods are, however, *not dense enough* or the new dense nonlinear mapping is *not equivalent to* f.

Suppose $f(\boldsymbol{x}; \boldsymbol{w})$ is a nonlinear mapping specified by parameters $\boldsymbol{w}$, which is implemented by multiple layers in a CNN. For another nonlinear mapping applied to the input $\boldsymbol{x}$, we say f and g is ***equivalent*** if f and g will *lead to the same output if the parameters are the same*, i.e., $f(\boldsymbol{x}; \boldsymbol{w}) = g(\boldsymbol{x}; \boldsymbol{w})$. Obviously, an equivalent nonlinear mapping g has to have the same (or equivalent) network architecture as f, and the parameters of g must have the same size and format as that of f. Compared to non-equivalent ones, an equivalent mapping has the following advantages:

equivalent的优势：
1. 转换现有模型以进行密集预测。
2. 反向传输缓慢密集的网络参数以改善快速原始CNN。
3. 模块化架构和更广泛的应用领域。

Transform existing models to have dense predictions. Many existing CNN models can be directly utilized, e.g., as an excellent initialization for fine-tuning the new dense prediction models. A simple approach for dense equivalent mapping is to set the stride of the *last* pooling layer to 1 or "shift and merge" [17, 15, 22]. However, if the stride in the last pooling layer was s in the mapping f, setting the stride to 1 in the new mapping g only increases the number of predictions by s^2 times. Setting the stride to 1 for more pooling layers will increase the number of predictions, but will lead to non-equivalent mappings [1].

dense相等映射g允许学习更好的参数集，因为更多的训练实例，评估概率和groundtruth信息被利用来学习g，

Inverse transfer a slow dense net's parameters back to improve the fast original CNN. More importantly, a dense equivalent mapping g allows learning a better set of parameters $\hat{\boldsymbol{w}}$, because more training instances, estimated probabilities and information in the groundtruth are utilized in learning the parameters of g. Because of the equivalence, we can *directly apply the parameters* $\hat{\boldsymbol{w}}$ *to* f, and $f(\boldsymbol{x}; \hat{\boldsymbol{w}})$ is *a better CNN model than the original CNN* $f(\boldsymbol{x}; \boldsymbol{w})$! Dense computation also means that g has higher computational cost than f. However, $f(\boldsymbol{x}; \hat{\boldsymbol{w}})$ has the same computational complexity as the original model $f(\boldsymbol{x}; \boldsymbol{w})$, but its accuracy is close to that of the dense mapping $g(\boldsymbol{x}; \hat{\boldsymbol{w}})$, as will be shown by our experiments. To the best of our knowledge, this paper is the first to transfer the parameters of a dense computation model back to the original model, which strikes a balance between high accuracy and affordable computational cost.

相同计算量

相同的精度

本文首次提出迁移dense网络的参数到原来的模型，在高准确率和计算开销上的平衡。

Modular architecture and wider application domains. We achieve dense predictions by introducing two new layers into existing CNN models: the equivalent convolution (eConv) and equivalent pooling (ePool) layers, and a new hyper-parameter: equivalent stride (eST). The dilated convolution layer [1, 23] effectively increases the receptive field size, which has achieved high accuracy in semantic segmentation. However, dilated convolution leads to non-equivalent mappings. The dilated convolution is the same as our eConv layer, but we can obtain equivalent mappings if the proposed ePool layers replace ordinary pooling layers. With the combination of eConv and ePool, we can produce dense predictions or estimations in CNN models for semantic segmentation, recognition (classification), localization and age estimation (regression) tasks, and our experiments show that these dense predictions are effective in improving the accuracy of all these models. Turning eConv and ePool layers back into ordinary convolution and pooling layers (but with their better parameters) lead to CNN models with higher accuracy but the same complexity as their respective original models.

我们实现密集预测通过两个已有的CNN模型：eConv和ePool层，和一个超参数：eST (equivalent stride);

The rest of this paper is organized as follows. We introduce the details of eConv and ePool layers in Section 2, which also includes a comparison of these layers with related works. Empirical validation results and relevant discussions are presented in Section 3. Section 4 ends this paper with concluding remarks and discussions for future work.

2 Dense equivalent computation using eConv and ePool layers

Equation 1 illustrates a classic CNN model with convolution, pooling and ReLU layers, in which the variables in different layers are denoted as x, y, z, s and t, respectively. Classic convolution and

pooling layers are illustrated in Figure 1a. Without padding, a 3×3 convolution layer reduces the output size by 2 and a pooling layer halves the output size in both directions. In Figure 1, we use the coordinates of the top-left element involved in a convolution or pooling operation to denote the output, e.g., $\left[\begin{smallmatrix} y_{55} & y_{56} \\ y_{65} & y_{66} \end{smallmatrix}\right]$ is max pooled to form z_{55} in Figure 1a.

$$\boxed{x} \xrightarrow[\text{ReLU}\\ \text{Layer 1}]{\text{Conv (3x3, st=1)}} \boxed{y} \xrightarrow[\text{Layer 2}]{\text{Max Pool}\\ \text{(2x2, st=2)}} \boxed{z} \xrightarrow[\text{ReLU}\\ \text{Layer 3}]{\text{Conv (3x3, st=1)}} \boxed{s} \xrightarrow[\text{Layer 4}]{\text{Max Pool}\\ \text{(2x2, st=2)}} \boxed{t} \xrightarrow[\text{ReLU}\\ \text{Layer 5}]{\text{Conv (3x3, st=1)}} \cdots \tag{1}$$

$$\boxed{x} \xrightarrow[\text{ReLU}\\ \text{Layer 1}]{\text{Conv (3x3, st=1)}} \boxed{y} \xrightarrow[\text{Layer 2}']{\text{Max Pool}\\ \text{(2x2, st=1)}} \boxed{z} \xrightarrow[\text{eST=2, ReLU}\\ \text{Layer 3}']{\text{eConv (3x3, st=1)}} \boxed{s} \xrightarrow[\text{Layer 4}']{\text{Max ePool}\\ \text{(2x2, st=1), eST=2}} \boxed{t} \xrightarrow[\text{eST=4, ReLU}\\ \text{Layer 5}']{\text{eConv (3x3, st=1)}} \cdots \tag{2}$$

Equation 2 shows the structure of a dense equivalent mapping for the CNN specified by Equation 1. The stride of pooling layers are all set to 1 (layer $2'$ and $4'$). After the first pooling layer, convolution and pooling layers are changed to eConv and ePool layers with appropriate eST values. The equivalent stride is 1 initially, but is multiplied by s after we change the stride of a pooling layer from s to 1.

2.1 The equivalent convolution and equivalent pooling layer

Figure 1b illustrates two changes between the architectures in Equation 2 and 1: setting the stride to 1 in one pooling layer, and an equivalent convolution layer (eConv) after it.

The output of a stride 2 pooling has only half size as its input in both directions, but a stride 1 pooling almost does not reduce the output size. Hence, denser predictions are available by reducing the stride size. However, as shown in Figures 1a and 1b, further applying a convolution layer will lead to non-equivalent mappings. In Figure 1a, s_{11} is convolved using $z_{ij}, i, j \in \{1, 3, 5\}$, but a classic convolution operator applied to Figure 1b will generate s_{11} using $z_{ij}, i, j \in \{1, 2, 3\}$!

The solution to correct this non-equivalence is to add an equivalent stride (eST) in the next convolution to make them equivalent. Let $\boldsymbol{z} \in \mathbf{R}^{H \times W \times D}$ be the input tensor to the convolution, $\boldsymbol{s}$ be the convolution result, and $\boldsymbol{f} \in \mathbb{R}^{h \times w \times D}$ be one convolution kernel, the equivalent convolution is simply

$$s_{ij} = \sum_{1 \le i' \le h} \sum_{1 \le j' \le w} \sum_{1 \le d \le D} f_{i',j',d} \times z_{i+(i'-1) \times eST, j+(j'-1) \times eST, d} \,. \tag{3}$$

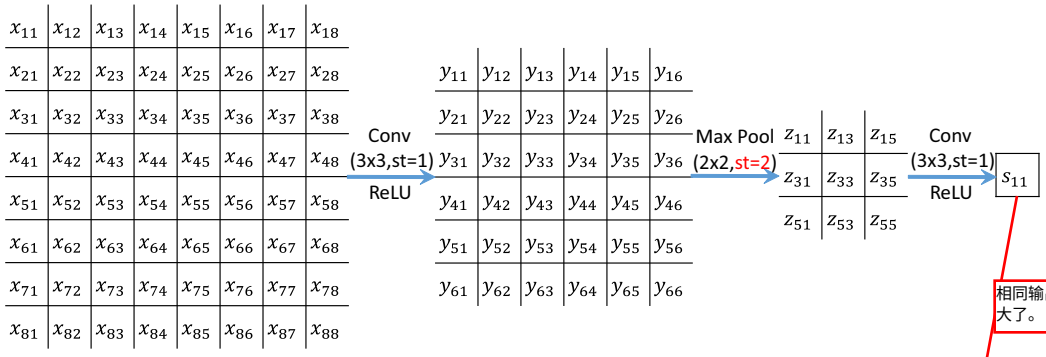
Because the current eST is 2, it is clear that now the s_{11} element will have the same value (i.e., be equivalent) in Figures 1a and 1b, since it is convolved using these z elements in blue circles. The equivalence holds for all elements that appear in Figure 1a, i.e., s_{ij} when both i and j are odd numbers. The other s_{ij} elements in Figure 1b are the new dense prediction results. For example, s_{22} (shown in the red circle) is computed from the receptive field $x_{ij}, 2 \le i, j \le 9$, which is the red square in the input image.

The subsequent layer is an equivalent pooling layer (illustrated in Figure 1c). In the original network, s_{11}, s_{13}, s_{31} and s_{33} are max-pooled to form t_{11} (cf. Figure 1a, some elements are not shown in that figure). A max pooling using an equivalent stride of 2 (cf. the circled elements in the blue dashed square in Figure 1c) produces t_{11} using exactly the same computations. An equivalent max pooling with size $h \times w$ is defined by

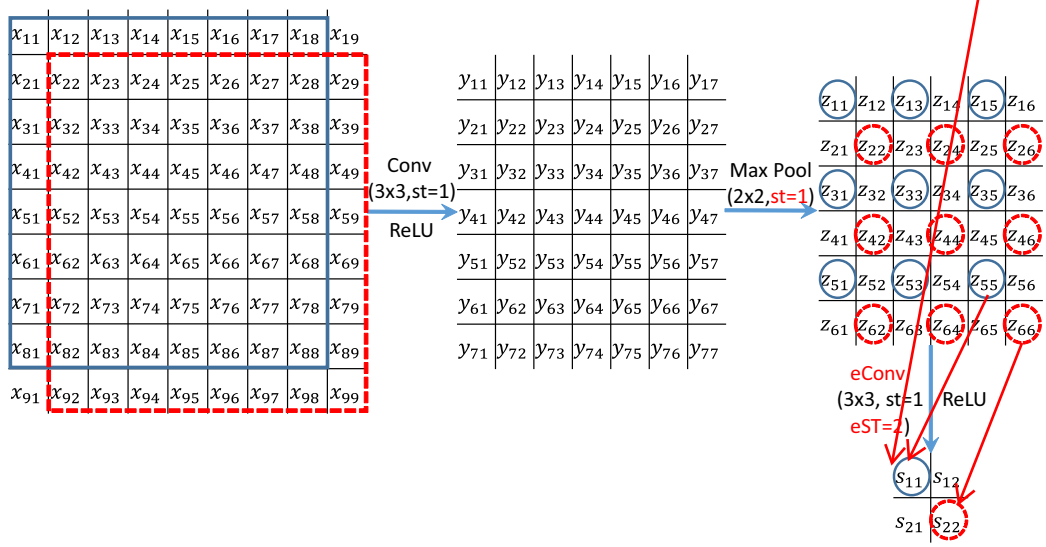
$$t_{i,j,d} = \max_{\substack{1 \le i' \le h \\ 1 \le j' \le w}} s_{i+(i'-1) \times eST, j+(j'-1) \times eST, d} \,. \tag{4}$$

The two dashed blue squares denote two max-pooling operations performed in the original network. Hence, after this equivalent pooling layer, the equivalent stride should be 4 (multiplied by 2) to maintain the equivalence.

As illustrated in Figure 1, the eConv and ePool layers (when equipped with appropriate eST values for different layers) maintain the equivalent mapping property with the original network, but produces much denser predictions (outputs). It is easy to observe that ReLU, batch normalization and fully connected layers (when implemented as convolution/eConv) will not invalidate the equivalence between the nonlinear mappings. Hence, the proposed eConv and ePool layers can turn many existing CNN models into an equivalent model and can output dense predictions.

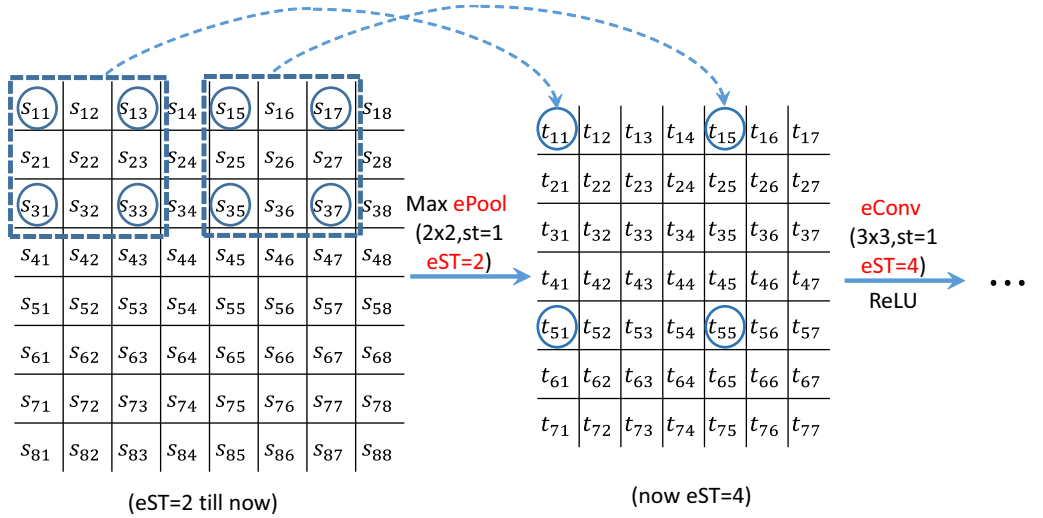


(a) Illustration of the classic convolution and pooling layers



(b) After a classic pooling layer with stride 1, the equivalent convolution layer requires an equivalent stride 2 to obtain the same (equivalent) computations as the network in Figure (1a)

和空洞卷积很相似。



(c) The equivalent pooling layer (whose eST=2) increase the equivalent stride for subsequent layers to eST=4

Figure 1: Illustration of the equivalence between classic convolution/pooling layers with eConv/ePool layers. This figure is best viewed in color.

We implement the eConv layer by rewriting the `im2col` and `col2im` functions in Caffe [6], and write a new ePool layer. These implementations enable both the forward pass and error back propagation in CNN learning for the new dense prediction net.

2.2 Related methods

The proposed equivalent convolution layer has the same computations (Equation 3) as the dilated convolution kernel [23, 1]. However, without the proposed equivalent pooling layer (Equation 4), dilated convolution leads to a non-equivalent computation to the original model. Dilated convolution can incorporate information from pixels in a large context, but only from a sparse subset of pixels in the receptive filed. Dilated convolution has been successfully applied in semantic segmentation [23, 1, 13]. The pooling layers after the first dilated convolution layer are removed (or stride set to 1) in these models. The proposed eConv and ePool layers utilize all pixels in the receptive field. A recent work removed the last two max pooling layers and increased last few convolution kernel size to 5×5 and 25×25 [11]. Large convolution kernels are, however, computationally expensive.

If one changes the last pooling layer's stride to 1 (or by shifting and merging several pooling results), both denser predictions and equivalent nonlinear mappings are achieved [17, 15, 22]. However, this strategy maintains equivalence only if there is no convolution or pooling layer after the changed pooling layer. Hence, the increasing in the number of predictions is small (usually $2^2 = 4$ or $3^2 = 9$). The proposed eConv and ePool layers can replace all (or any subset) of classic convolution and pooling layers in a CNN model, so long as the equivalent stride is set accordingly. If proper padding is added, eConv and ePool can produce prediction at every pixel location (i.e., 500×500 predictions for a 500×500 input image). However, to reduce the memory and CPU/GPU usage, we use classic convolution and pooling in the first few layers, and change later ones to eConv and ePool.

An early work in medical image segmentation implemented dense equivalent mappings [3]. The fast dense scan in [3] was implemented by generating multiple *fragments* after every pooling layer, in which s^2 fragments are generated for a pooling layer with stride s. The network in [3] had 4 pooling layers and $s = 2$; hence, 256 fragments were generated in the end. The strategy in [3] supports forward computation, but not error back propagation. The proposed eConv and ePool layers not only generate dense predictions, but also support back propagation to learn better parameters.

3 Experimental results

We use the proposed eConv and ePool layers to replace classic convolution and pooling layers in various CNN models. Because of the equivalence, the new model is initialized with the parameters of the original model. This initialization is also the final model in unsupervised tasks. Fine-tuning is used in supervised learning tasks to learn better parameters for the CNN. We call the new network the `dense` net. When we assign the dense net's parameters to the original network, we call it the `dpoa` (dense net's paramters, original net's architecture) net.

We show that dense nets are effective in various tasks, including at least semantic segmentation, object localization, recognition, and age estimation. We mainly use the FCN or VGG-16 model as the baseline CNN network, and present the empirical validation results in this section.

3.1 Semantic segmentation

Semantic segmentation is a natural application of the proposed eConv and ePool layers. A semantic segmentation system needs to predict the semantic category (e.g., horse, person, bicycle etc.) of every pixel in the input image. Hence, dense prediction is a desired property for such systems.

The baseline CNN we adopt is the fully convolutional network (FCN) [12]. The FCN-32s model outputs 16×16 predictions for a 500×500 input image, which is upsampled to predict for every pixel location. The number 32 means the upsampling ratio. By a skip connection between the fourth pooling layer (`pool4`) and the output, the prediction resolution becomes 34×34, which is called the FCN-16s model. Similarly, a skip connection from `pool3` leads to the FCN-8s model, whose output is the densest (70×70) and the most accurate within the FCN family.

The proposed `dense` net changes the `pool3` layer's stride to 1 and converts subsequent convolution and pooling layers to eConv and ePool, respectively. Hence, the number of predictions in the `dense`

Table 1: IoU comparison on Pascal VOC 2011 and VOC 2012 [2].

(a) 2011 validation

	IoU		IoU
FCN-32s	59.4%	`dense-32s`	62.4%
FCN-16s	62.4%	`dense-16s`	64.8%
FCN-8s	62.7%	`dense-8s`	65.3%

(b) 2011 test

	IoU
FCN-8s	62.7%
`dense-8s`	66.7%
`dpoa-8s`	65.6%

(c) 2012 validation

	IoU
FCN-8s	61.3%
`dense-8s`	66.2%
`dpoa-8s`	63.9%

(d) 2012 test

	IoU
FCN-8s	62.2%
`dense-8s`	68.4%
`dpoa-8s`	66.5%

Table 2: Pascal VOC 2012 test set IoU comparison. The baseline CNN is FCN-8s.

	plane	bike	bird	boat	bott	bus	car	cat	chair	cow	tbl	dog	horse	motor	person	plant	sheep	sofa	train	tv	mean
FCN-8s	76.8	34.2	68.9	49.4	60.3	75.3	74.7	77.6	21.4	62.5	46.8	71.8	63.9	76.5	73.9	45.2	72.4	37.4	70.9	55.1	62.2
`dense`	**83.0**	**38.6**	**74.1**	**58.6**	**65.7**	**83.2**	**81.5**	**80.7**	**29.2**	**73.9**	**53.8**	**74.8**	**73.0**	**79.3**	**78.3**	**56.3**	**80.2**	**46.2**	**72.7**	**61.0**	**68.4**
`dpoa`	79.4	35.0	72.4	55.0	65.5	82.3	78.8	79.4	27.1	71.3	53.1	74.3	70.8	77.9	75.9	53.5	79.0	42.9	71.3	59.3	66.5

net is $64(= 8 \times 8)$ times as many as that in FCN. As in FCN, we use `dense-`x`s` to denote the net learned with different skip connections, $x \in \{8, 16, 32\}$. The parameters in a `dense-32s` net are initialized from FCN-32s, and fine-tuned using the training set (e.g., Pascal VOC in this section.) The `dense-16s` net is fine-tuned from `dense-32s`, and `dense-8s` is fine-tuned from `dense-16s`.

VOC 2011. We first tested on the Pascal VOC 2011 dataset, using the *same* training and validation images as the original FCN [12]. The training set include 8825 images (1112 from VOC2011 and the rest are extended ones from [4]); the validation set has 736 images. The mean class-wise intersection over union (IoU) score is the main metric to evaluate semantic segmentation results, and a higher IoU means better segmentation. The background is considered as a class in computing the mean IoU. We tested FCN, `dense` and `dpoa` nets on both the validation set and the VOC 2011 test server.

VOC 2012. We also tested on the Pascal VOC 2012 dataset using the same training and validation images as those in [24]. The training set includes 11685 images (1464 from VOC 2012 and the rest are extended ones from [4]); the validation set has 346 images. We tested both FCN and `dense` nets on both the validation set and the VOC 2012 test server. Results on both VOC 2011 and VOC 2012 are shown in Table 1.

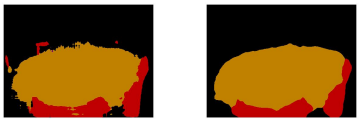
The `dense` nets are more accurate than baseline FCNs. As shown in Tables 1a to 1d, the proposed `dense` nets are more accurate than their corresponding FCN nets. The smallest improvement is 2.4%, between FCN-16s and `dense-16s` on the VOC 2011 validation set. On the test set and VOC 2012, the improvement is much larger. For example, the mean IoU of `dense-8s` is 6.2% higher than that of FCN-8s on the VOC 2012 test set!

The `dpoa` net is a good compromise between speed and accuracy. The `dpoa` net has exactly the same architecture and speed as the baseline FCN net. As shown in Tables 1b and 1d, the `dpoa` net has 2.8% and 4.3% higher mean IoU than baseline FCN-8s on the VOC 2011 and VOC 2012 test sets, respectively. Its accuracy is much closer to the `dense` net than FCN. Hence, when fast testing speed is required, the proposed `dpoa` net is very useful.

The increase of training examples helps `dense` and `dpoa`. We show the per-category IoU on the VOC 2012 test set in Table 2. In *every* category, we always observe the same ranking in terms of IoU: `dense`>`dpoa`>FCN, which further supports the effectiveness of both proposed networks. However, we observe that the improvement is small in categories with thin structures (e.g., bicycle and chair), but is large for objects with bulk structures (e.g., cow and potted-plant). We conjecture that the higher IoU of `dense` is because more training examples and labels are used due to the increased number of predictions. The performance of `dpoa` also supports this conjecture. It has the same architecture as FCN, hence its higher IoU means that its parameters are better than that in FCN, which is the consequence of training with more pixels (denser predictions) and their labels.

The `dense` and `dpoa` nets may be further improved by adding CRF and more training images. The conditional random field (CRF) has been shown to boost the accuracy of semantic segmentation methods by a large margin [1, 22–24]. Although we did not experiment with adding CRF on top of

(a) Input (b) Groundtruth (c) FCN-8s



(d) `dense-8s` (e) `dpoa-8s`

Figure 2: Example semantic segmentation results of different models.

(a) VGG-16 net (b) `dense` net

Figure 3: Unsupervised object localization using the VGG-16 net (3a) and `dense` net (3b).

the proposed nets, we expect this combination will lead to higher segmentation accuracy. As shown in Figure 2, `dense-8s` detects more details of the chairs than FCN-8s or `dpoa-8s`. However, it also contains more non-smooth predictions, such as the yellow pixels within the group of red pixels (i.e., predicting a table inside a chair). CRF is an expert in correcting this kind of errors. The usage of additional training data, e.g., MS COCO (`http://mscoco.org/`) should be useful, too.

3.2 Unsupervised object localization

eConv and ePool layers are effective in unsupervised object localization, too. Wei et al. [21] proposed an unsupervised method to localize the main object in fine-grained datasets, e.g., the CUB200-2011 dataset [20]. They showed that the sum of `pool5` output ($7 \times 7 \times 512$, across the channel axis) of VGG-16 is a good indicator to localize the main object. The sum (7×7 in size) is truncated and converted to a binary mask using their average value as the threshold, and resized to the input image size as a mask to localize the object. Figure 3a shows one example of the localization, including the input image and the mask (shown in red), and the localized object in the bounding box computed from the mask (enlarged in the figure). Using the 50% IOU criterion, the localization accuracy is 76.09% in our implementation (76.79% in [21]) on the CUB200-2011 dataset.

We change the stride of `pool4` to 1 and change subsequent layers to eConv and ePool in the VGG-16 net. Since this is an unsupervised object localization method, no fine-tuning is performed. As shown in Figure 3b, this `dense` net can localize the bird with tighter boundaries. The localization accuracy is 79.06% using the 50% IOU criterion. The dense net has achieved a 3% improvement over the VGG-16 net. Because no new parameters are obtained, a `dpoa` net is not applicable for this task. As in the semantic segmentation, eConv and ePool layers provide denser predictions than the original network, hence more accurate localization is obtained, although the same parameters are used in the `dense` net and the VGG-net.

3.3 Image categorization

Image categorization is a classic application of CNN [8, 7]. We tested the `dense` net on the scene categorization dataset `indoor` [16], which has 67 scene classes (roughly 80 training and 20 testing images per category). The baseline net is VGG-16 fine-tuned on the training images. The input image is resized to 256×256 and random crops of size 224×224 are used for training. 256×256 images

are used directly for testing, yielding 2×2 probability estimates. The final prediction is based on the average of them. In the `dense` net, we change the `pool4` stride to 1 and subsequent layers to eConv and ePool, which generates 31×31 probability estimates. We add a global average pooling layer to turn them into one estimate. The `dense` net is initialized with the baseline model's parameters and fine-tuned using the training images.

On the `indoor` dataset, the fine-tuned VGG-16 net's accuracy is 73.06%, `dense` net has 73.51% accuracy, and the `dpoa` net achieves 74.02% accuracy. The `dense` net has higher accuracy than VGG-16, but the `dpoa` net has the highest among them. Because image categorization is not a pixel labeling task, we conjecture that averaging a large number of dense predictions is not as effective as directly applying the better parameters to the original CNN architecture.

We also evaluated the testing speed on the `indoor` dataset. The running time of VGG-16, `dpoa` and `dense` are 44ms, 44ms and 132ms per image, respectively. As expected, the testing speed of `dpoa` is as fast as that of VGG-16. As for `dense`, although it produces 240 times more predictions, its speed is only 2 times slower than that of the baseline.

3.4 Apparent age estimation

We also tested the `dense` net on the apparent age estimation problem, which tries to estimate the age of a person based on a single picture. The dataset is the ChaLearn 2015 competition data, which has 2476 training and 1136 validation images.[1] We start from the VGG-Face model [14]. Given an input image, the face within it is detected, cropped and aligned to the frontal pose. It is then resized to 224×224. The groundtruth age a is converted into the range $[-1 \ \ +1]$ by $\frac{a-45}{45}$. We add a $\tanh$ and an ℓ_1 loss layer to VGG-Face, and use the ChaLearn training images to fine-tune and get the baseline CNN model (which we still call VGG-Face). To train the `dense` net, we change the stride of VGG-Face's `pool4` layer to 1 and change subsequent layers to eConv and ePool. The last fully connected layer will output 27×27 predictions. We add a global average pooling layer to summarize them, and also add a $\tanh$ and an ℓ_1 loss layer. The `dense` net is initialized with the baseline model's parameters and fine-tuned using the training images.

The evaluation uses the validation set, and the metric is mean absolute error (MAE) (i.e., number of years difference between the groundtruth and the prediction). The MAE of VGG-Face, `dense` and `dpoa` are 6.82, 5.28 and 6.19 years, respectively. Both `dense` and `dpoa` have smaller errors than the baseline net. This time `dense` has a large winning margin over `dpoa` (0.91 year, or 15% relative improvement). Our conjecture is: because the face alignment cannot be perfect, the ensemble of a lot of predictions has an advantage in compensating for the inaccurate alignments.

4 Conclusions and discussions

In this paper, we introduced two layers for CNN: equivalent convolution (eConv) and equivalent pooling (ePool). These layers can turn many existing CNN models into new networks that output dense predictions. The same (equivalent) nonlinear mapping is applied to different receptive fields in the input image, including those already computed by the original model and many more.

The equivalence allows learning the dense model starting from existing CNN models, and more importantly, allows transferring better parameters in the dense model back to the baseline CNN model and improving its accuracy. This inverse transfer allows both fast speed of the original CNN net and higher accuracy. The proposed layers are not only useful for applications requiring dense pixel labeling, but many others. Experiments on semantic segmentation, unsupervised object localization, image categorization and age estimation verified the effectiveness of the proposed layers.

An obvious future research direction is to test the proposed eConv and ePool layers in more CNN architectures, e.g., CNN+CRF in semantic segmentation (e.g., [10]) and deeper CNN architectures such as the ResNet [5] and GoogLeNet [19]. The proposed dense prediction layers are also expected to improve other tasks requiring dense pixel labeling, such as object detection without resorting to region proposals, depth estimation from single color image, saliency detection, scene understanding for automatic (autonomous) driving and intrinsic image decomposition. It is also potentially useful for learning recurrent CNN (e.g., [9]).

[1] http://gesture.chalearn.org/2015-looking-at-people-iccv-challenge

References

[1] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. In *International Conference on Learning Representations*, pages 1–14, 2015.

[2] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015.

[3] A. Giusti, D. C. Cireşan, J. Masci, L. M. Gambardella, and J. Schmidhuber. Fast image scanning with deep max-pooling convolutional neural networks. In *ICIP*, pages 4034–4038, 2013.

[4] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik. Semantic Contours from Inverse Detectors. In *ICCV*, pages 991–998, 2011.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[6] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. In *ACM Multimedia*, pages 675–678, 2014.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS 25*, pages 1097–1105, 2012.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[9] M. Liang, X. Hu, and B. Zhang. Convolutional Neural Networks with Intra-Layer Recurrent Connections for Scene Labeling. In *NIPS 28*, pages 937–945, 2015.

[10] G. Lin, C. Shen, I. Reid, and A. van den Hengel. Deeply Learning the Messages in Message Passing Inference. In *NIPS 28*, pages 361–369, 2015.

[11] Z. Liu, X. Li, P. Luo, C. C. Loy, and X. Tang. Semantic Image Segmentation via Deep Parsing Network. In *ICCV*, pages 1377–1385, 2015.

[12] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *CVPR*, pages 3431–3440, 2015.

[13] P. Ondrúška, J. Dequaire, D. Z. Wang, and I. Posner. End-to-End Tracking and Semantic Segmentation Using Recurrent Neural Networks. *arXiv preprint arXiv:1604.05091*, pages 1–9, 2016.

[14] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep Face Recognition. In *British Machine Vision Conference*, pages 1–12, 2015.

[15] P. O. Pinheiro and R. Collobert. Recurrent Convolutional Neural Networks for Scene Labeling. In *ICML*, pages 82–90, 2014.

[16] A. Quattoni and A. Torralba. Recognizing Indoor Scenes. In *CVPR*, pages 413–420, 2009.

[17] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv preprint arXiv:1312.6229v4*, pages 1–16, 2014.

[18] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[19] C. Szcgcdy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. In *CVPR*, pages 1–9, 2015.

[20] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD birds-200-2011 dataset. Technical Report CNS-TR-2011-001, Caltech Computation & Neural Systems, 2011.

[21] X.-S. Wei, J.-H. Luo, and J. Wu. Selective Convolutional Descriptor Aggregation for Fine-Grained Image Retrieval. *arXiv preprint arXiv:1604.04994*, pages 1–16, 2016.

[22] Z. Wu, C. Shen, and A. van den Hengel. High-performance Semantic Segmentation Using Very Deep Fully Convolutional Networks . *arXiv preprint arXiv:1604.04339*, pages 1–11, 2016.

[23] F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. In *International Conference on Learning Representations*, pages 1–9, 2016.

[24] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional Random Fields as Recurrent Neural Networks . In *ICCV*, pages 1529–1537, 2015.