LPRNet: Lightweight Deep Network by Low-rank Pointwise Residual Convolution

Bin Sun, Jun Li, Ming Shao, Yun Fu

Abstract

Deep learning has become popular in recent years primarily due to the powerful computing device such as GPUs. However, deploying these deep models to end-user devices, smartphones, or embedded systems with limited resources is challenging. To reduce the computation and memory costs, we propose a novel lightweight deep learning module by low-rank pointwise residual (LPR) convolution, called LPRNet. Essentially, LPR aims at using low-rank approximation in pointwise convolution to further reduce the module size, while keeping depthwise convolutions as the residual module to rectify the LPR module. This is critical when the low-rankness undermines the convolution process. We embody our design by replacing modules of identical input-output dimension in MobileNet and ShuffleNetv2. Experiments on visual recognition tasks including image classification and face alignment on popular benchmarks show that our LPRNet achieves competitive performance but with significant reduction of Flops and memory cost compared to the state-of-the-art deep models focusing on model compression.

Introduction

During past years, deep convolutional neural networks (DCNN) have been widely used in many areas of machine learning and computer vision, such as face synthesis (Dollar, Welinder, and Perona 2010), image classification (He et al. 2016), pose estimation (Cao et al. 2017), and many more. However, the model complexity of DCNN in terms of time and space makes it hard for direct applications on mobile and embedded devices (Howard et al. 2017; Sandler et al. 2018; Zhang et al. 2018; Ma et al. 2018). In fact, the depthwise convolution applies a single filter to each input channel, and the pointwise convolution only uses $1 \times 1$ filters to compute the output features. Thus, the number of the parameters in the DSC module is significantly reduced to $mk^2 + nm$. It should be noted that most of the popular DCNN models still use a large number of channels for better performance. Therefore, the pointwise convolution in DSC or relevant models still suffer from the higher time and space complexity. Besides, some modules like MobileNetv2 cannot fit into the existing networks without changing their architectures. Thus, a lighter model targeting at these problems is our research goal in this paper which may promote the deployment of more DCNNs to mobile applications.

To address the above problems, we introduce a novel DCNN parameters reduction module inspired by the principle of the low-rank CP-decomposition method (Jaderberg, Vedaldi, and Zisserman 2014; Lebedev et al. 2015). Instead of decomposing learned weight matrices, we apply the CP-decomposition on the layer design. Besides decomposing the conventional full-channel convolution into depthwise and pointwise convolutions, we will develop new learning
paradigms for each of them, and thus reduce the overall model complexity. Following low-rank matrix decomposition idea, we are motivated to divide the large pointwise convolution into two small low-rank pointwise convolutions, as shown in Figure 5. When its rank is $r < n, m$, the number of the parameters is further reduced to $mk^2 + (n + m)r$. Furthermore, to compensate for the low-rankness of pointwise convolution and performance recession due to this compression, a residual operation through depthwise convolution is implemented to complement the feature maps without any additional parameters. To summarize, the contributions of this paper are:

- We propose a low-rank pointwise residual (LPR) module by automatically decomposing a larger pointwise convolution module into two low-rank matrices through the network learning, which significantly reduces the computational consumption.

- A residual learning mechanism is implemented to compensate for the information loss in pointwise convolution due to the matrix low-rankness, which guarantees the performance of our lightweight model without additional cost.

- We embed our designed module in the network structure of MobileNet and ShuffleNetv2, and validate that our module can significantly reduce the parameters and FLOPs while keeping the performance with the same architecture.

- We also validate the correctness on image classification and face alignment tasks. On ImageNet dataset, while using much smaller parameters compared to the state-of-the-art, we achieved very competitive performance. On challenging face alignment benchmarks, our LPRNet obtains comparable results.

Related Work

In this section, we will first review the related work on the lightweight network construction. Then an overview of the state-of-the-art on image classification will be given. Last some related work on face alignment will be presented.

Deep lightweight Structure

In recent years, some methods have been emerged for speeding up the deep learning model. Faster activation function named rectified-linear activation function (ReLU) was proposed to accelerate the model (Glorot, Bordes, and Bengio 2011). Jin et al. (Jin, Dundar, and Culurciello 2014) showed the flattened CNN structure to accelerate the feedforward procedure. In (Sifre and Mallat 2014) depthwise separable convolution was initially introduced and was used in Inception models (Szegedy et al. 2015). Xception network (Chollet 2017), MobileNet (Howard et al. 2017), Sandler et al. (Sandler et al. 2018), and ShuffleNet (Zhang et al. 2018) used group Lasso to constrain the structure scale of LRA in CNN.

To utilize DNN structure to different databases, Feng et al. (Feng and Darrell 2015) learned the appropriate number of filters in DNN. Wen et al. (Wen et al. 2016) applied group Lasso to regulate multiple DNN structures. In 2017, an architecture termed SVDNet (Sun et al. 2017) also considered matrix low-rankness in their framework to optimize the deep representation learning process. Different from SVDNet, our LPRNet utilizes the low-rank strategy as the guidance in the structure construction.

Image Classification

Image classification has been extensively used to evaluate the performance of different deep learning models. For example, small-scale datasets (Krizhevsky and Hinton 2009) and large-scale datasets (Deng et al. 2009a, Deng et al. 2009b) are often adopted as benchmarks in state-of-the-art works. In 2012, AlexNet was invented and considered as the first breakthrough DCNN model on ImageNet (Krizhevsky, Sutskever, and Hinton 2012). Simonyan et al. later presented a deep network called VGG (Simonyan and Zisserman 2015) which further boosted the state-of-the-art performance on ImageNet. GoogLeNet (Szegedy et al. 2015) presented better results via an even deeper architecture. What followed is the widely adopted deep structure termed ResNet (He et al. 2016) which enabled very deep networks and presented the state-of-the-art in 2016. Huang et al. further improved ResNet by densely using residuals in different layers, called Densenet (Huang et al. 2017) and improved the performance on ImageNet in 2017. Inception-v4 (Szegedy et al. 2017) is a novel structure that embraces the merits of both ResNet and GoogLeNet. While our LPRNet development is based on low-rank matrix decomposition, we also use residual term to compensate information loss due to compression. Most importantly, it retains the performance while reducing the parameters and computational burden.

Face Alignment

In conventional face alignment works, patch based regression methods were widely discussed (Cootes et al. 1995; Baltrušaitis, Robinson, and Morency 2012; Cootes, Edwards, and Taylor 2001; Baltrušaitis, Robinson, and Morency 2016) in past decades. In addition, tree-based methods (Kazemi and Sullivan 2014; Ren et al. 2014) with plain features attracted more attention, and achieved high speed alignment. Based on optimization theory, a cascade of weak regressors was implemented for face alignment (Xiong and De la Torre 2013). Along with the rise of deep learning, Sun et al. (Sun, Wang, and Tang 2015) firstly utilized CNN model for face alignment with a face image as the input to CNN module, followed by regression on high-level features. It spawned considerable deep models (Trigeorgis et al. 2016; Liu et al. 2017; Zhu et al. 2016; Sun et al. 2018a; Bhagavatula et al. 2017; Ranjan, Patel, and Chellappa 2017; Kumar and Chellappa 2018) that achieved good results on large pose face alignment. Besides, recently published large pose face alignment datasets with 3D warped faces for large poses (Zhu et al. 2016), or DNN structure Glass (Yang, Liu, and Zhang 2017; Bulat and Tzimiropoulos 2017) have significantly promote
the development and benchmarks in this field. We also evaluate our LPRNet on large pose face alignment problem to show its effectiveness and efficiency on the regression tasks, and impressive performance on accuracy, speed, and size will be demonstrated in experiment section.

**LPRNet**

In this section, we will elaborate the proposed LPRNet. First, we introduce the standard convolution (Simonyan and Zisserman 2015) and depthwise separable convolution from a matrix products perspective (Howard et al. 2017). Next, we propose our novel LPR structure and use it as building block in LPRNet. Finally, discussions and preliminary experiments of the LPRNet are shown. Notations of this paper have been summarized in Table 1.

| Variable | Description |
|----------|-------------|
| $S_F$    | size of one feature map |
| $S_K$    | size of one kernel |
| $C_{in}$ | number of input channels |
| $C_{out}$| number of output channels |
| $D_{ij}$ | weight of the $i$-th feature in Depthwise layer |
| $p_{ij}$ | weight of the $j$-th filter in Pointwise layer |
| $W_{ij}$ | weight of the $j$-th feature in Standard Conv. |
| $F_j$    | $j$-th feature map of the input |

Table 1: Notations summary.

**Standard Convolutions (SConv)**

In traditional DCNNs, the convolution operation is applied between each filter and the input feature map. Essentially, the filter applies different weights to different features while doing convolution. Afterwards, all features convoluted by one filter will be added together to generate a new feature map. The whole procedure is equivalent to a series of matrix products, which can be formally written as:

$$
\begin{bmatrix}
W_{i1} & \ldots & W_{i m} \\
\vdots & \ddots & \vdots \\
W_{n1} & \ldots & W_{nm}
\end{bmatrix} \otimes \begin{bmatrix} F_1 & F_2 & \ldots & F_m \end{bmatrix}^T,
$$

where $W_{ij}$ is the weight of the $i$-th filter corresponding to the $j$-th feature map, $F_j$ is the input feature map, and $W_{ij} \otimes F_j$ means the feature map $F_j$ is convoluted by filter with the weight $W_{ij}$. In this paper, each $W_{ij}$ is a $3 \times 3$ matrix (filter), and all of them constitute a large matrix $[W_{ij}]$, or simply $W$.

**Depthwise Separable Convolutions (DSC)**

Depthwise Separable Convolution layers are the keys for many lightweight neural networks (Zhang et al. 2018; Howard et al. 2017; Sandler et al. 2018). It has two layers: depthwise convolutional layer and pointwise convolutional layer (Howard et al. 2017).

Depthwise convolutional layer applies a single convolutional filter to each input channel which will massively reduce the parameter and computational cost. Following the process of its convolution, we can describe the depthwise convolution using in the form of matrix products:

$$
\begin{bmatrix} D_{11} & \ldots & 0 \\
\vdots & \ddots & \vdots \\
0 & \ldots & D_{mn}
\end{bmatrix} \otimes \begin{bmatrix} F_1 & F_2 & \ldots & F_m \end{bmatrix}^T
$$

in which $D_{ij}$ is usually a $3 \times 3$ matrix, and $m$ is the number of the input feature maps. We define $D$ as the matrix $[D_{ij}]$. Since $D$ is a diagonal matrix, the depthwise layer has much less parameters than a standard convolution layer.

**Pointwise convolutional layer** is using $1 \times 1$ convolution to build the new features through computing the linear combinations of all input channels. It follows the fashion of traditional convolution layer with the kernel size set to 1. Following the process of its convolution, we can describe the pointwise convolution in the form of matrix products:

$$
\begin{bmatrix} p_{11} & \ldots & p_{1m} \\
\vdots & \ddots & \vdots \\
p_{n1} & \ldots & p_{nm}
\end{bmatrix} \otimes \begin{bmatrix} F_1 & F_2 & \ldots & F_m \end{bmatrix}^T
$$

in which $p_{ij}$ is a scalar, $m$ is the number of the input feature maps, and $n$ is the number of the output. The computational cost is $S_F \times S_F \times C_{in} \times C_{out}$, and the number of parameters is $C_{in} \times C_{out}$. We define $P \in \mathbb{R}^{m \times n}$ as the matrix $[p_{ij}]$. Since the depthwise separable convolution composed with depthwise convolution and pointwise convolution, it can be represented as:

$$
F_{out} = W \otimes F_{in} \cong (P \times D) \otimes F_{in}
$$

**LPR Structure**

In this subsection, we will detail the proposed LPR module, which has been shown in Figure 3(c). Recall in the previous section, the depthwise convolution can be considered as the convolution between a diagonal matrix $\text{diag}(D_{11}, \ldots, D_{mm})$ and a feature map matrix $[F_1, \ldots, F_m]$. We will keep this procedure but further explore the pointwise convolution in the following manner. To further reduce the size of matrix $P$, we will pursue a low-rank decomposition of $P$ such that $P \approx P^{(2)} \times P^{(1)}$, $P^{(1)} \in \mathbb{R}^{r \times m}$, $P^{(2)} \in \mathbb{R}^{m \times r}$, $r \ll m$. Clearly, the highest rank of this approximation is $r$, and the size of $m \times r$ is much smaller than $m^2$. Thus, we can naturally convert the original DSC module to:

$$
F_{out} = (P^{(2)} P^{(1)}) \otimes F_{in}^D,
$$

where $F^{P}$ means the output features after this new low-rank pointwise convolution operation. While using the strategy above may reduce the parameters and computational cost, it may undermine the original structure of $P$ when $r$ is inappropriately small, e.g., $r < \text{rank}(P)$. To address this issue, we propose to add a term $F_{res} = D \otimes F_{in}$, i.e., the original feature map after the depthwise convolution with $D$. This ensures that if the overall structure of $P$ is compromised, the depthwise convolution is still able to capture the spatial features of the input. Interestingly, this is conceptually similar to the popular residual learning where $F_{in}$ is added to the module output, but ours uses $D \otimes F_{in}$ instead. By considering this residual term, we can finally formulate our low-rank pointwise residual module as:

$$
(P \times D) \otimes F_{in} \cong F^{P} + F_{res} = (P^{(2)} P^{(1)} + I_m) D \otimes F_{in}.
$$
where $I_m$ is an identity matrix. To further improve the performance, we may normalize the features of $F_m^{(1)}$ with L2 Normalization on the channel, and apply batch normalization on $D$. With the factorization of the large matrix $P$, our LPR successfully reduces the parameters and computational costs comparing with other state-of-the-art modules.

### Ablation Study

In this subsection we will first show the experiment of $r$ selection. Then an ablation study of our LPR module will be presented. At last we will validate our low-rank approach with an experiment on CIFAR-10.

| Rank of LPR | ImageNet Top-1 | FLOPs | Parameters |
|-------------|-----------------|-------|------------|
| $r = m/4$   | 70.8%           | 303M  | 3.0M       |
| $r = m/8$   | 70.0%           | 260M  | 2.3M       |
| $r = m/16$  | 68.1%           | 209M  | 2.0M       |
| MobileNet   | 70.0%           | 574M  | 4.2M       |

Table 3: Experiments to select the best rank $r$. $m$ means the number of the output channels.

A set of experiments on ImageNet with MobileNet architecture has been performed to select the best rank $r$ during the low-rank decomposition which is shown in Equation (6). The results are shown in Table 3. From the table we can find that $r = m/8$ keep good performance while leads to significant reduction on the computational cost and parameters. With $r = m/8$ as the rank control parameter, the theoretical comparisons among the prevalent lightweight modules are shown in Table 2. In this table, our LPR module has the least computational cost and parameters than ShuffleNetv2 module. Thus, $r$ should be smaller than $m/4$. Note that $4r < m − S_k^2$ is the sufficient and necessary condition which can make LPR module have less computational cost and parameters than ShuffleNetv2 module. Thus, $r$ should be smaller than $m/4$. Note that $P_{mr}^{(1)}$ and $P_{mr}^{(2)}$ are learned to approximate the optimized matrices through training.

To validate the effectiveness of different parts in our LPR module, we train our LPRNet on the CIFAR-10 datasets after remove the L2 Normalization layer and residual part respectively. The comparison results are shown in the Table

| Models | Accuracy | Parameters |
|--------|----------|------------|
| SConv (Simonyan and Zisserman 2015) | 92.1% | 11M |
| DSC Module (Howard et al. 2017) | 91.8% | 3.1M |
| LPR Module(no Residual) | 88.5% | 2.4M |
| LPR Module(no L2Norm) | 90.0% | 2.4M |
| LPR Module | 91.8% | 2.4M |

Table 4: Performance with different modules. Since the parameters are fixed during the training, the only updated modules are DSC and LPR. Therefore, the similar accuracy among different modules means the weight matrices are similar.

### Figure 2: The heatmap visualization of the difference among standard Convolution, DSC, and LPR. The similarity is represented by MSE. The lower MSE means the two feature matrices have higher similarity. Better viewed in color.

From the table, it is clear that the completed LPR module has similar performance with the DSC module. However, its performance will drop after we remove the residual part. Besides, the performance will suffer a significant recession after we remove the L2 Normalization layer. Both parts will not increase the parameters of the model.

We also design an experiment to verify the ability of the Low-Rank approach of our LPR module. In our experiment, a network using standard convolution is trained on CIFAR-10. Then we replace a layer whose dimension is $512 \times 512 \times 14 \times 14$ with DSC module, LPR without Residual, LPR without L2 Normalization, and our LPR module.
**Table 5:** Performance of image classification task on ImageNet. The first region shows direct comparisons between the LPRNet and its underlying structures; the rest regions are divided based on the size of parameters (from smallest to largest).

### Implementations

In this subsection, we embody the LPRNet based on our LPR module and the deep learning structure of MobileNetv1 and ShuffleNetv2, respectively. The reason for choosing MobileNet [Howard et al. 2017] and ShuffleNetv2 [Ma et al. 2018] is that most modules of these two networks have the same input dimension and output dimension, which is the condition to utilize our LPR module. The details of the module we used in our LPRNet are shown in Figure 5. Since the input of the LPR module requires identical input/output dimension, the downsample modules of the MobileNetv1 and ShuffleNetv2 are reserved in our LPRNet. The rest modules are replaced by our LPR module.

### Experiment

In this section, we conduct experiments on image classification and large poses Face Alignment tasks. In each subsection, we present datasets, comparison methods, parameter settings, evaluation metrics, and show comparison results.

### Image Classification

**Dataset:** To make an fair comparison, the dataset we use is ImageNet 2012 classification dataset [Deng et al. 2009a; Russakovsky et al. 2015]. There are 128,117 images and 1,000 classes in training dataset. The images in training dataset are resized to 480 × 480 and are randomly cropped. The images in validation dataset are resized to 256 × 256 and are center cropped. Some augmentations such as random flip, random scale, and random illumination are implemented on the training dataset. All the results are tested on the validation dataset.

**Comparison Methods:** In this section, we compare with
lightweight architectures on ImageNet classification including MobileNetv1 (Howard et al. 2017), ShuffleNetv1 (Zhang et al. 2018), MobileNetv2 (Sandler et al. 2018), ShuffleNetv2 (Ma et al. 2018), MnasNet (Tan et al. 2018), CondenseNet (Huang et al. 2018), IGCV3 (Sun et al. 2018b), and ESP-Netv2 (Mehta et al. 2019).

Parameter Settings: Our learning model is built by Mxnet framework (Chen et al. 2015). The optimizer is the large batch SGD (Bottou 2010) starting with the learning rate 0.5. The learning rate is decayed following cosine function. The total epoch number is set to 210 for LPR-MobileNet, and 400 for LPR-ShuffleNetv2. The batch size is set to 256 for LPR-MobileNet and 400 for LPR-ShuffleNetv2. After training, the model is tuned on the same training dataset without data augmentation.

Evaluation Metrics: In this paper, we evaluate the performance using Top-1 accuracy. Like other works (Sandler et al. 2018; Ma et al. 2018; Tan et al. 2018; Yang et al. 2018), the computational cost is evaluated using calculated FLOPs number and the parameters are evaluated by the calculated number.

Comparison Results: The comparison results are shown in the Table 5. The table is divided into six regions. The first region is the direct comparison between the LPRNet and its underlying architecture from MobileNetv2 and ShuffleNetv2. The rest regions are divided based on parameters. Compared with the same architecture, our LPR-MobileNet can retain the same performance while reduce the computational cost and parameters significantly. Though the LPR-ShuffleNetv2 has 0.9% a lower accuracy than ShuffleNetv2, it is only 85% of the original size and uses 75% fewer Flops than ShuffleNetv2. Compared with other methods, the LPRNet also achieves the best performance with approximately the same complexity. When the parameters are reduced to the K level, our LPRNet has over 63% Top-1 accuracy while the accuracy of all other methods is below 57%. Even the parameters are reduced to the least, we can still achieve 50% accuracy. The MobileNetv2×0.25 and ShuffleNetv2×0.5 are not shown in the K level since their parameters are larger than 1M.

Large Poses Face Alignment

Datasets: In our face alignment experiments, all the baselines use 68-point landmarks to conduct fair comparisons. We evaluate all the baselines with only x-y coordinates for fair comparisons, since some datasets (Bulat and Tzimiropoulos 2017) used only have 2D coordinates projected from 3D landmarks. Training datasets are 300W-LP (Zhu et al. 2016), while testing datasets are AFLW2000-3D (Zhu et al. 2016), Re-annotated AFLW2000-3D (Zhu et al. 2016), Re-annotated AFLW2000-3D (Bulat and Tzimiropoulos 2017), LS3D-W (Bulat and Tzimiropoulos 2017) which has 5 sub-dataset Menpo-3D (8,955 images), 300W-3D (600 images), and 300VW-3D (A, B, and C).

Comparison Methods: We conduct comprehensive evaluations with the state-of-the-art methods. In this section, a comparison is made with state-of-the-art deep methods including PCD-CNN (Kumar and Chellappa 2018), 3DFAN (Bulat and Tzimiropoulos 2017), Hyperface (Ranjjan, Patel, and Chellappa 2017), 3DSTN (Bhagavatula et al. 2017), 3DDFA (Zhu et al. 2016), and MDM (Trigeorgis et al. 2016). Among these baselines, the results of 3DSTN and PCD-CNN are cited from the original papers. We also compare the accuracy and speed on CPU, with some those methods only running on CPU, including SDM (Xiong and De la Torre 2013), ERT (Kazemi and Sullivan 2014), and ESR (Cao et al. 2014). To make a fair comparison, the lightweight models MobileNet (Howard et al. 2017; Sandler et al. 2018) and ShuffleNet (Zhang et al. 2018; Ma et al. 2018) are implemented for face alignment and are trained on the same datasets. All these models are using half channels for fast training and testing.

Parameter Settings: Our structure is built by Mxnet framework (Chen et al. 2015) and uses $L_2$ loss specified for regression task. We use Adam stochastic optimization (Kingma and Ba 2014) with default hyper-parameters to learn the weights. The initial learning rate is set to 0.0005, and the
| Method Name               | NME (°) | Mean NME (°) | FPS | Memory (Bytes) |
|--------------------------|---------|--------------|-----|----------------|
| LPRNet_{MobileNet} ×0.25 | 2.58    | 3.33         | 25  | 619K           |
| LPRNet_{MobileNet} ×0.5  | 2.41    | 3.15         | 619 | 1.6M           |
| ShuffleNetV2_{0.5}       | 2.54    | 3.32         | 52  | 3.3M           |
| ShuffleNetV2_{0.5}       | 3.03    | 4.01         | 104 | 2.1M           |
| MobileNetV2_{0.5}        | 2.51    | 3.15         | 40  | 3.3M           |
| MobileNetV1_{0.5}        | 2.52    | 3.21         | 42  | 2.5M           |
| PCD-CNN_{0.5}            | 2.93    | 4.14         | -   | -              |
| Hyperface_{TPAMI}        | 3.93    | 4.14         | 119.7M | 119.7M |
| 3DSTN_{0.5}              | 3.15    | 3.98         | 52  | -              |
| 3DFAN_{0.5}              | 2.75    | 3.76         | 183M| -              |
| 3DDFA + SDM_{0.5}        | 3.43    | 4.24         | 121M| -              |
| MDM_{0.5}                | 3.67    | 5.94         | 307M| -              |
| ERT_{0.5}                | 5.40    | 7.12         | 95M | -              |
| ESR_{0.5}                | 4.60    | 6.70         | 248M| -              |
| SDM_{0.5}                | 3.67    | 4.94         | 10M | -              |

Table 6: Comparisons with state-of-the-art methods on AFLW2000-3D dataset.

---

**Evaluation Metrics:** We use ground-truth landmarks to generate bounding boxes. “Normalized Mean Error (NME)” is an important metric for face alignment evaluation, which is defined as $\text{NME} = \frac{1}{N} \sum_{i=1}^{N} \frac{\| \hat{X}_i - X_i \|}{d}$, where $\hat{X}$ and $X$ are predicted and ground truth landmarks, respectively, and $N$ is the number of the landmarks. $d$ can be computed using $d = \sqrt{w_{bbox} \times h_{bbox}}$, in which $w_{bbox}$ and $h_{bbox}$ mean the width and height of the bounding box, respectively. The speed of all methods is evaluated on Intel-i7 CPU without Openmp. We use Frames Per Second (FPS) to evaluate the speed. The storage size in this paper is calculated from the compressed model.

**Comparison Results:** To compare the performance of the different range of angles, we divide the testing dataset into three parts by the range of the angles of the faces (Zhu et al. 2016). The curve of the cumulative errors distribution (CED) of the whole dataset is shown in Figure 4. The visual comparison results are shown in Figure 5. From the Table 6 we can observe that the NME of our LPRNet ×0.5 is 5% lower than the current state-of-the-art PCD-CNN. The NME of our LPRNet ×0.25 achieves similar performance as PCD-CNN and MobileNetv1. Comparison with other lightweight model is also shown in the table. Comparing with MobileNetv1 and MobileNetv2, our LPRNet ×0.25 has similar NME but with ×1.8 speed on CPU and 73% compression ratio.

**Time Complexity:** Compared with those traditional deep learning methods (Ranjan, Patel, and Chellappa 2017; Bhagavatula et al. 2017; Zhu et al. 2016; Bulat and Tzimiropoulos 2017; Hinton, Vinyals, and Dean 2015; Trigeorgis et al. 2016), our LPRNet has much better speed on both one core CPU and GPU. In the table, we notice that the SDM (Xiong and De la Torre 2013), ERT (Kazemi and Sullivan 2014) and ESR (Cao et al. 2014) have very impressive speed on CPU. The reason is all of these methods use hand-craft features, which are easy to compute by computers but have limited ability for representation. Comparing with the light weight models (Howard et al. 2017; Sandler et al. 2018; Zhang et al. 2018; Ma et al. 2018), our LPRNet achieves the second best speed on CPU when the parameters are downscaled by 0.25, but get much better performance than the fastest model (Zhang et al. 2018).

**Space Complexity:** For the applications on mobile devices, the memory size of the model should be smaller enough. From the Table 6 it can be observed that our LPRNet is ×120 smaller than the smallest model in baseline deep learning methods except for MobileNet. Besides, it is ×3.4 smaller than the smallest model ShuffleNetv1 with much lower NME.

**Conclusion**

In this paper, we proposed a novel lightweight deep learning module named LPR to further reduce the network parameters through low-rank matrix decomposition and residual learning. By applying our LPR module to MobileNet and ShuffleNetv2, we managed to reduce the size of existing lightweight models. We surprisingly found that on image classification and face alignment tasks, compared to many state-of-the-art deep learning models, our LPRNet had much lower parameters and computational cost, but kept very competitive or even better performance. More importantly, this casts a light on deep models compression through low-rank matrix decomposition, and enables many powerful deep models to be deployed in end devices. We plan to release our models (in Mxnet) upon the publication of this work.
