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, smart phones, 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 (Dollár, 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). Therefore, there is an urge to design dedicated DCNN modules to reduce the computational cost and storage size for further applications on end devices. Furthermore, to make full use of existing networks, a general and efficient module is desired to replace the standard convolution module without changing the architectures.

Standard convolution operation in Fig. 1 (a) includes a large number of parameters (i.e., nmk^2, where $k \geq 3$ is the size of filters, n and m are the numbers of output and input feature channels or maps), which results in a high time and space complexity. To reduce the complexities, the standard convolution is divided into depthwise and pointwise convolutions, namely, depthwise separable convolution (DSC) (Sifre and Mallat 2014; Howard et al. 2017) in Fig. 1 (b). Based on the DSC module, many lightweight networks are demonstrated, such as Xception model (Chollet 2017), SqueezeNet (Iandola et al. 2017), MobileNet (Howard et al. 2017; Sandler et al. 2018), ShuffleNet (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 1×1 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

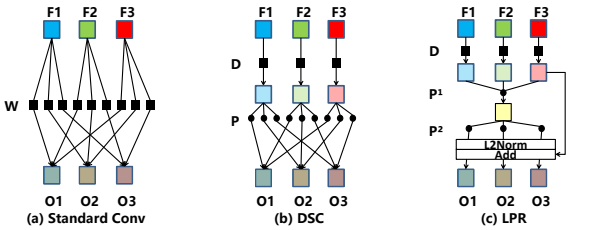

Figure 1: Illustration of light convolution architectures. (a): standard convolution. (b): depthwise separable convolution (DSC) module. (c): our low-rank pointwise residual (LPR) module.

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 3 (c). 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) show the flattened CNN structure to accelerate the feed-forward procedure. In (Sifre and Mallat 2014) depthwise separable convolution was initially introduced and was used in Inception models (Ioffe and Szegedy 2015), Xception network (Chollet 2017), MobileNet (Howard et al. 2017; Sandler et al. 2018), and ShuffleNet (Zhang et al. 2018; Ma et al. 2018), condensenet (Huang et al. 2018).

Group Lasso (Yuan and Lin 2006) is an efficient regularization for learning sparse structures. Jaderberg et.al. (Jaderberg, Vedaldi, and Zisserman 2014) implemented the low-rank theory on the weights of filters with the separate convolution in different dimensions. Liu et al. (Liu et al. 2015) implemented 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 regularize 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 DensenNet (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 2013) 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

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}	ith weight for the jth feature in Depthwise layer
p_{ij}	ith weight for the jth feature in Ponitwise layer
W_{ij}	ith weight for the jth feature in Standard Conv.
$\otimes$	Kronecker Product
F_j	jth feature map of the input

Table 1: Notations summary.

the development and benchmarks in this field. We also evalute our LPRNet on large pose face alignment problem to show its effectiveness and efficiency on the regression tasks, and mpressive performance on accuracy, speed, and size will 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 in Table 1.

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:

$$\begin{bmatrix} W_{11} & \dots & W_{1m} \\ \vdots & \ddots & \vdots \\ W_{n1} & \dots & W_{nm} \end{bmatrix} \otimes \begin{bmatrix} F_1 & F_2 & \dots & F_m \end{bmatrix}^T, \tag{1}$$

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 3×3 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:

$$\begin{bmatrix} D_{11} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & D_{mm} \end{bmatrix} \otimes \begin{bmatrix} F_1 & F_2 & \dots & F_m \end{bmatrix}^T \tag{2}$$

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

Pointwise convolutional layer is using 1×1 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:

$$\begin{bmatrix} p_{11} & \dots & p_{1m} \\ \vdots & \ddots & \vdots \\ p_{n1} & \dots & p_{nm} \end{bmatrix} \otimes \begin{bmatrix} F_1 & F_2 & \dots & F_m \end{bmatrix}^T \tag{3}$$

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 $S_{\mathrm{F}} \times S_{\mathrm{F}} \times C_{in} \times C_{out}$, 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 searable convolution composed with depthwise convolution and pointwise convolution, it can be represented as:

$$F_m^{out} = W \otimes F_m^{\mathrm{in}} \approx (P \times D) \otimes F_m^{\mathrm{in}} \tag{4}$$

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 $dig(D_{11}, ...D_{mm})$ and a feature map matrix $[F_1...F_m]$. 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:

$$F_m^P = (P_{mr}^{(2)} P_{rm}^{(1)}) \otimes F_m^D, \tag{5}$$

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., $r < \mathrm{rank}(P)$. To address this issue, we propose to add a term $F_m^{Res} = D \otimes F_m^{in}$, 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_m^{in} is added to the module output, but ours uses $D \otimes F_m^{in}$ instead. By considering this residual term, we can finally formulate our low-rank pointwise residual module as:

$$(P \times D) \otimes F_m^{\mathrm{in}} \approx F_m^P + F_m^{Res} = (P_{mr}^{(2)} P_{rm}^{(1)} + I_m) D_m \otimes F_m^{in}, \tag{6}$$

Modules	computational cost (FLOPs)	Parameters
SConv (Simonyan and Zisserman 2015)	$S_{\mathrm{k}}^2 \times S_{\mathrm{F}}^2 \times C_{in} \times C_{out}$	$S_{\mathrm{k}}^2 \times C_{in} \times C_{out}$
DSC (Howard et al. 2017)	$(\frac{1}{C_{out}} + \frac{1}{S_{\mathrm{k}}^2}) \times S_{\mathrm{k}}^2 \times S_{\mathrm{F}}^2 \times C_{in} \times C_{out}$	$(\frac{1}{C_{out}} + \frac{1}{S_{\mathrm{k}}^2}) \times S_{\mathrm{k}}^2 \times C_{in} \times C_{out}$
Mobilev2 (Sandler et al. 2018)	$(\frac{e}{C_{out}} + \frac{e+1}{S_{\mathrm{k}}^2}) \times S_{\mathrm{k}}^2 \times S_{\mathrm{F}}^2 \times C_{in} \times C_{out}$	$(\frac{e}{C_{out}} + \frac{e+1}{S_{\mathrm{k}}^2}) \times S_{\mathrm{k}}^2 \times C_{in} \times C_{out}$
Shufflev2 (Ma et al. 2018)	$\frac{1}{2}(\frac{1}{C_{out}} + \frac{1}{S_{\mathrm{k}}^2}) \times S_{\mathrm{k}}^2 \times S_{\mathrm{F}}^2 \times C_{in} \times C_{out}$	$\frac{1}{2}(\frac{1}{C_{out}} + \frac{1}{S_{\mathrm{k}}^2}) \times S_{\mathrm{k}}^2 \times C_{in} \times C_{out}$
LPR	$(\frac{1}{C_{out}} + \frac{2}{kS_{\mathrm{k}}^2}) \times S_{\mathrm{k}}^2 \times S_{\mathrm{F}}^2 \times C_{in} \times C_{out}$	$(\frac{1}{C_{out}} + \frac{2}{kS_{\mathrm{k}}^2}) \times S_{\mathrm{k}}^2 \times C_{in} \times C_{out}$
$S_{\mathrm{F}} = 14, S_{\mathrm{k}} = 3, C_{in} = 256, C_{out} = 256, k = 8, e = 4$		
SConv (Simonyan and Zisserman 2015)	$115,605,504$	$589,824$
DSC (Howard et al. 2017)	$13,296,640$	$67,840$
Mobilev2 (Sandler et al. 2018)	$66,031,616$	$336,896$
Shufflev2 (Ma et al. 2018)	$6,648,320$	$33,920$
LPR	$3,662,848$	$18,688$

Table 2: Comparisons on Flops and parameters. SConv, DSC, Shufflev2, Mobilev2, and LPR modules are used to build VGG, Mobilenetv1, Mobilenetv2, ShuffleNetv2 and our LPRNet respectively.

where I_m is an identity matrix. To further improve the performance, we may normalized the features of $F_m^{(P)}$ 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%	363M	3.0M
$r = m/8$	70.6%	260M	2.3M
$r = m/16$	68.1%	209M	2.0M
MobileNet	70.6%	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 when the input and output are the same. Note that $4r < m - S_{\mathrm{k}}^2$ 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 $P_{mr}^{(2)}$ and $P_{rm}^{(1)}$ 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 4. 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.

Models	Accuracy	Paramters
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)	90.6%	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.

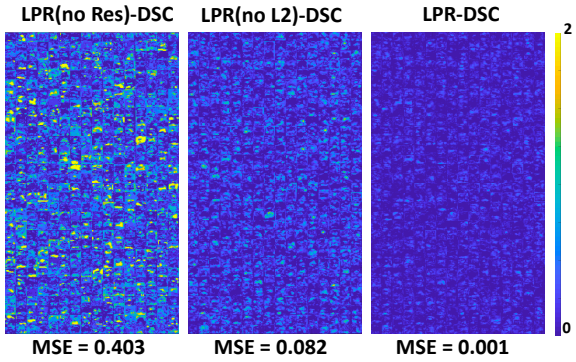

Figure 2: The heatmap visualization of the difference among standard Convolution, DSC, and LPR. The similarity is represented by MSE. The lower MSE means the two feature matrices have higher similarity. Better viewed in color.

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

Methods	Top-1	FLOPs	params	Methods	Top-1	FLOPs	params
ShuffleNetv2$_{2018ECCV}$×1.0 (Ma et al. 2018)	69.3%	149M	2.3M	MobileNetv1$_{2017\ CoRR}$×1.0 (Howard et al. 2017)	70.6%	574M	4.2M
LPRNet$_{Shufflev2}$×1.0	69.1%	113M	2.0M	**LPRNet**$_{MobileNet}$×1.0	70.6%	260M	2.3M
ShuffleNetv1$_{2018CVPR}$×0.25 (Zhang et al. 2018)	38.5%	13M	368K	MobileNetv1$_{2017CoRR}$×0.25 (Howard et al. 2017)	50.6%	42M	470K
ShuffleNetv2$_{2018ECCV}$ ×0.25 (Ma et al. 2018)	43.0%	14M	587K	ShuffleNetv1$_{2018CVPR}$×0.5 (Zhang et al. 2018)	56.8%	41M	718K
LPRNet$_{MobileNet}$×0.25	50.1%	26M	356K	**LPRNet**$_{MobileNet}$×0.5	63.2%	78M	869K
IGCV3$_{2018BMVC}$×0.5 (Sun et al. 2018b)	60.6%	111M	2.0M	ESPNetv2$_{2019CVPR}$×1.0 (Mehta et al. 2019)	64.6%	98M	1.7M
MobileNetv1$_{2017CoRR}$×0.5 (Howard et al. 2017)	63.7%	152M	1.3M	ESPNetv2$_{2019CVPR}$×1.25 (Mehta et al. 2019)	66.8%	138M	1.9M
MobileNetv2$_{2018CVPR}$ ×0.5 (Sandler et al. 2018)	64.3%	100M	1.9M	**LPRNet**$_{Shufflev2}$×1.0	68.4%	113M	2.0M
ShuffleNetv1$_{2018CVPR}$×1.0 (Zhang et al. 2018)	67.4%	140M	2.2M	ShuffleNetv2$_{2018ECCV}$×1.0 (Ma et al. 2018)	69.3%	149M	2.3M
ESPNetv2$_{2019CVPR}$×1.5 (Mehta et al. 2019)	67.9%	185M	2.3M	CondenseNet$_{2018CVPR}$ (Huang et al. 2018)	70.3%	291M	2.9M
IGCV3$_{2018BMVC}$×0.75 (Sun et al. 2018b)	69.1%	210M	2.6M	**LPRNet**$_{MobileNet}$×1.0	70.6%	260M	2.3M
MobileNetv1$_{2017CoRR}$×0.75 (Howard et al. 2017)	68.4%	325M	3.6M	IGCV3$_{2018BMVC}$×1.0 (Sun et al. 2018b)	71.7%	340M	3.4M
ESPNetv2$_{2019CVPR}$×2.0 (Mehta et al. 2019)	71.0%	306M	3.5M	MobileNetv2$_{2018CVPR}$×1.0 (Sandler et al. 2018)	72.0%	301M	3.5M
ShuffleNetv1$_{2018CVPR}$×1.5 (Zhang et al. 2018)	71.5%	292M	3.4M	**LPRNet**$_{MobileNet}$×1.25	72.8%	389M	3.4M
MobileNetv1$_{2017\ CoRR}$×1.0 (Howard et al. 2017)	70.6%	574M	4.2M	ShuffleNetv2$_{2018ECCV}$×2.0 (Ma et al. 2018)	74.1%	595M	7.6M
ShuffleNetv1$_{2018CVPR}$×2.0 (Zhang et al. 2018)	73.4%	524M	5.4M	MobileNetv2$_{2018CVPR}$×1.4 (Sandler et al. 2018)	74.4%	585M	6.9M
LPRNet$_{Shufflev2}$×2.0	73.8%	437M	6.2M	**LPRNet**$_{MobileNet}$×1.5	74.6%	544M	4.5M

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).

respectively. The network is trained while all the other parameters are fixed. Training process is stopped when the model has similar Top-1 validation accuracy with the original network. The similarities among output features are represented by Mean Squire Error (MSE) and visualized by the heatmaps. The results are shown in Figure 2. As shown in the figure, the MSE between LPR and DSC is only 0.001, which means the output features from LPR and DSC have little difference. Since the other parameters of the network are fixed, similar output features mean the weights of the two modules are similar, which supports that the weight matrix of our low-rank decomposition structure can approach the matrix of Depthwise Separable Convolution. We can also find from the figure that the MSE increases when we remove the residual part and the L2 Normalization. Furthermore, the MSE increases more than 400 times when the residual part is removed from our LPR module, which indicates that the learned weights is hard to approach the weights of the DSC module.

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 3. Since the input of the LPR module requires identical input-output dimension, the down sample modules of the MobileNetv1 and ShuffleNetv2 are reserved in our LPRNet. The rest modules are replaced by our LPR module.

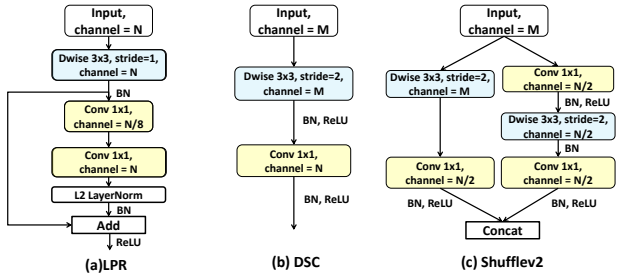

Figure 3: (a): Our LPR module. (b): Down-sample module using Depthwise Separable Convolution. (c): Down-sample module using ShuffleNetv2 module. Note that DSC is also used when the input and output of the module are not in the same dimension.

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 $128,1167$ 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 secion, we compare with

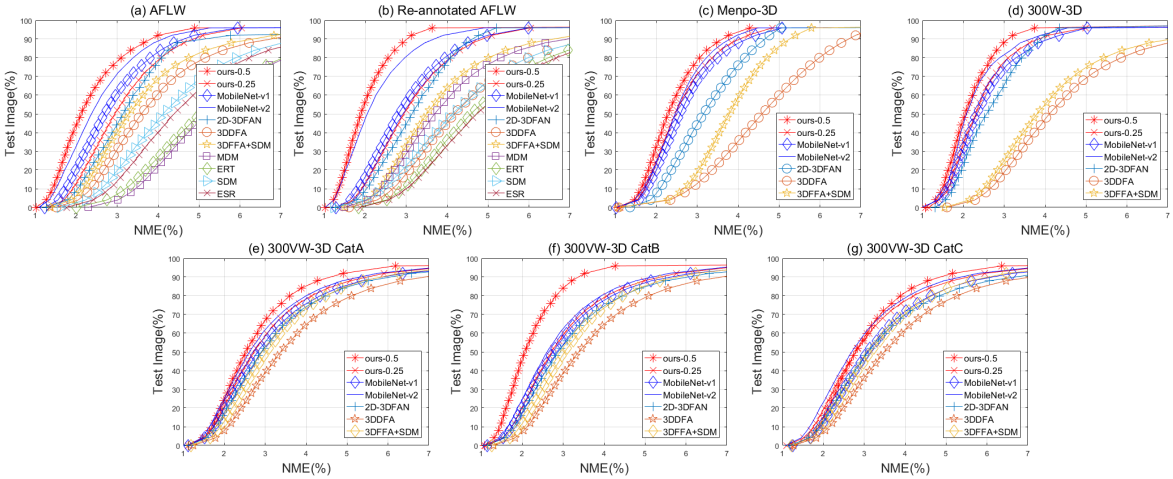

Figure 4: The CED curves on different test datasets.We test the 3D methods on whole datasets of AFLW2000-3D, re-annotated AFLW2000-3D, Menpo-3D, 300W-3D, and 300VW-3D. The top-left curve means the method has the best performance, i.e., most cases below given NME.

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 ESPNetv2 (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 $\text{LPR}_{\text{MobileNet}}$, and 400 for $\text{LPR}_{\text{Shufflev2}}$. The batch size is set to 256 for $\text{LPR}_{\text{MobileNet}}$ and 400 for $\text{LPR}_{\text{Shufflev2}}$. 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 $\text{LPR}_{\text{MobileNet}}$ can retain the same performance while reduce the computational cost and parameters significantly. Though the $\text{LPR}_{\text{Shufflev2}}$ has 0.9% a lower accuracy than ShuffleNetv2, it is only 83% 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$\times 0.25$ and ShuffleNetv2$\times 0.5$ are not shown in the K level since their parameters are largerh 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 (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 (Ranjan, 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

Normalized Mean Error on ALFW2000-3D					Speed (FPS)		Memory
Method Name	[0°30°]	[30°60°]	[60°90°]	Mean	GPU	CPU	params (Bytes)
LPRNet$_{\text{MobileNet}}$ $\times 0.25$	**2.58**	**3.33**	**5.81**	**3.89**	**800**	**95**	**619K**
LPRNet$_{\text{MobileNet}}$ $\times 0.5$	**2.41**	**3.15**	**5.52**	**3.69**	**630**	**52**	**1.6M**
ShuffleNetv2$_{\text{2018ECCV}}$ ×0.5 (Ma et al. 2018)	2.54	3.32	5.51	3.79	600	51	3.3M
ShuffleNetv1$_{\text{2018CVPR}}$×0.5 (Zhang et al. 2018)	3.03	4.01	6.07	4.37	900	104	2.1M
MobileNetv2$_{\text{2018CVPR}}$ ×0.5 (Sandler et al. 2018)	2.51	3.15	5.53	3.73	600	40	2.3M
MobileNetv1$_{\text{2017CoRR}}$×0.5 (Howard et al. 2017)	2.52	3.21	5.76	3.85	600	42	2.5M
PCD-CNN$_{\text{2018CVPR}}$ (Kumar and Chellappa 2018)	2.93	4.14	4.71	3.92	20	-	-
Hyperface$_{\text{2017TPAMI}}$ (Ranjan, Patel, and Chellappa 2017)	3.93	4.14	6.71	4.26	-	-	119.7M
3DSTN$_{\text{2017ICCV}}$ (Bhagavatula et al. 2017)	3.15	4.33	5.98	4.49	52	-	-
3DFAN$_{\text{2017ICCV}}$ (Bulat and Tzimiropoulos 2017)	2.75	3.76	5.72	4.07	6	< 1	183M
3DDFA$_{\text{2016CVPR}}$ (Zhu et al. 2016)	3.78	4.54	7.93	5.42	43	13	111M
3DDFA+SDM$_{\text{2016CVPR}}$ (Zhu et al. 2016)	3.43	4.24	7.17	4.94	31	9	121M
MDM$_{\text{2016CVPR}}$ (Trigeorgis et al. 2016)	3.67	5.94	10.76	6.45	5	< 1	307M
ERT$_{\text{2014CVPR}}$ (Kazemi and Sullivan 2014)	5.40	7.12	16.01	10.55	-	300	95M
ESR$_{\text{2014IJCV}}$ (Cao et al. 2014)	4.60	6.70	12.67	7.99	-	83	248M
SDM$_{\text{2013CVPR}}$ (Xiong and De la Torre 2013)	3.67	4.94	9.76	6.12	-	80	10M

Table 6: Comparisons with state-of-the-art methods on ALFW2000-3D dataset.

initialized weights are generated with Xavier initialization. The epoch is set to 60, and the batch size is set to 100. We set the learning rate to $4e^{-4}$ at first 15 epoch and then decay the learning rate to $2e^{-4}$ when the number of channels is multiplied by 0.5.

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 NME $= \frac{1}{N}\sum_{i=1}^{N}\frac{\left\|\hat{X}_i - X_i^*\right\|_2}{d}$ where the $\hat{X}$ and X^* is predicted and ground truth landmarks, respectively, and N is the number of the landmarks. d can be computed using $d = \sqrt{w_{\text{bbox}} \times h_{\text{bbox}}}$, 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 Openmpi. 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 visualization comparison results are shown in Figure 5. From the Table 6, we can observe that the NME of our LPRNet $\times 0.5$ is 5% lower than the current state-of-the-art PCD-CNN. The NME of our LPRNet $\times 0.25$ 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 $\times 0.25$ has similar NME but with $\times 1.8$ 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; 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 handcraft features, which are easy to compute by computers but

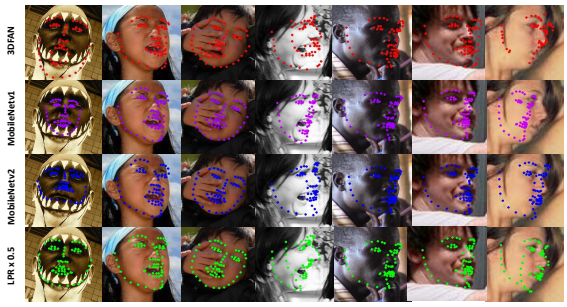

Figure 5: Some visualized results.

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 $\times 120$ smaller than the smallest model in baseline deep learning methods except for MobileNet. Besides, it is $\times 3.4$ 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.

References

[Baltrušaitis, Robinson, and Morency 2012] Baltrušaitis, T.; Robinson, P.; and Morency, L.-P. 2012. 3d constrained local model for rigid and non-rigid facial tracking. In *CVPR*, 2610–2617.

[Baltrušaitis, Robinson, and Morency 2016] Baltrušaitis, T.; Robinson, P.; and Morency, L.-P. 2016. Openface: an open source facial behavior analysis toolkit. In *WACV*, 1–10.

[Bhagavatula et al. 2017] Bhagavatula, C.; Zhu, C.; Luu, K.; and Savvides, M. 2017. Faster than real-time facial alignment: A 3d spatial transformer network approach in unconstrained poses. *ICCV* 2.

[Bottou 2010] Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, 177–186.

[Bulat and Tzimiropoulos 2017] Bulat, A., and Tzimiropoulos, G. 2017. How far are we from solving the 2d & 3d face alignment problem?(and a dataset of 230,000 3d facial landmarks). In *ICCV*, 1021–1030.

[Cao et al. 2014] Cao, X.; Wei, Y.; Wen, F.; and Sun, J. 2014. Face alignment by explicit shape regression. *IJCV* 107(2):177–190.

[Cao et al. 2017] Cao, Z.; Simon, T.; Wei, S.-E.; and Sheikh, Y. 2017. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*.

[Chen et al. 2015] Chen, T.; Li, M.; Li, Y.; Lin, M.; Wang, N.; Wang, M.; Xiao, T.; Xu, B.; Zhang, C.; and Zhang, Z. 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*.

[Chollet 2017] Chollet, F. 2017. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 1251–1258.

[Cootes et al. 1995] Cootes, T. F.; Taylor, C. J.; Cooper, D. H.; and Graham, J. 1995. Active shape models-their training and application. *CVIU* 61(1):38–59.

[Cootes, Edwards, and Taylor 2001] Cootes, T. F.; Edwards, G. J.; and Taylor, C. J. 2001. Active appearance models. *TPAMI* 23(6):681–685.

[Deng et al. 2009a] Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009a. Imagenet: A large-scale hierarchical image database. In *CVPR*, 248–255.

[Deng et al. 2009b] Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009b. Imagenet: A large-scale hierarchical image database. In *CVPR*, 248–255.

[Dollár, Welinder, and Perona 2010] Dollár, P.; Welinder, P.; and Perona, P. 2010. Cascaded pose regression. In *CVPR*, 1078–1085.

[Feng and Darrell 2015] Feng, J., and Darrell, T. 2015. Learning the structure of deep convolutional networks. In *ICCV*, 2749–2757.

[Glorot, Bordes, and Bengio 2011] Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep sparse rectifier neural networks. In *AISTATS*, 315–323.

[He et al. 2016] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

[Hinton, Vinyals, and Dean 2015] Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *NIPS*.

[Howard et al. 2017] Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR* abs/1704.04861.

[Huang et al. 2017] Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *CVPR*.

[Huang et al. 2018] Huang, G.; Liu, S.; van der Maaten, L.; and Weinberger, K. Q. 2018. Condensenet: An efficient densenet using learned group convolutions. In *CVPR*.

[Iandola et al. 2017] Iandola, F. N.; Han, S.; Moskewicz, M. W.; Ashraf, K.; Dally, W. J.; and Keutzer, K. 2017. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *ICLR*.

[Ioffe and Szegedy 2015] Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*.

[Jaderberg, Vedaldi, and Zisserman 2014] Jaderberg, M.; Vedaldi, A.; and Zisserman, A. 2014. Speeding up convolutional neural networks with low rank expansions. In *BMVC*.

[Jin, Dundar, and Culurciello 2014] Jin, J.; Dundar, A.; and Culurciello, E. 2014. Flattened convolutional neural networks for feedforward acceleration. *CoRR*.

[Kazemi and Sullivan 2014] Kazemi, V., and Sullivan, J. 2014. One millisecond face alignment with an ensemble of regression trees. In *CVPR*, 1867–1874.

[Kingma and Ba 2014] Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *ICLR*.

[Krizhevsky and Hinton 2009] Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

[Krizhevsky, Sutskever, and Hinton 2012] Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1097–1105.

[Kumar and Chellappa 2018] Kumar, A., and Chellappa, R. 2018. Disentangling 3d pose in a dendritic cnn for unconstrained 2d face alignment. *CVPR*.

[Lebedev et al. 2015] Lebedev, V.; Ganin, Y.; Rakhuba, M.; Oseledets, I.; and Lempitsky, V. 2015. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *ICLR*.

[Liu et al. 2015] Liu, B.; Wang, M.; Foroosh, H.; Tappen, M.; and Pensky, M. 2015. Sparse convolutional neural networks. In *CVPR*, 806–814.

[Liu et al. 2017] Liu, Y.; Jourabloo, A.; Ren, W.; and Liu, X. 2017. Dense face alignment. *ICCV Workshop*.

[Ma et al. 2018] Ma, N.; Zhang, X.; Zheng, H.-T.; and Sun, J. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 122–138.

[Mehta et al. 2019] Mehta, S.; Rastegari, M.; Shapiro, L.; and Hajishirzi, H. 2019. Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network. *CVPR*.

[Ranjan, Patel, and Chellappa 2017] Ranjan, R.; Patel, V. M.; and Chellappa, R. 2017. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *TPAMI*.

[Ren et al. 2014] Ren, S.; Cao, X.; Wei, Y.; and Sun, J. 2014. Face alignment at 3000 fps via regressing local binary features. In *CVPR*, 1685–1692.

[Russakovsky et al. 2015] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *IJCV* 115(3):211–252.

[Sandler et al. 2018] Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CVPR*.

[Sifre and Mallat 2014] Sifre, L., and Mallat, P. 2014. *Rigid-motion scattering for image classification*. Ph.D. Dissertation, Citeseer.

[Simonyan and Zisserman 2015] Simonyan, K., and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. *ICLR*.

[Sun et al. 2017] Sun, Y.; Zheng, L.; Deng, W.; and Wang, S. 2017. Svdnet for pedestrian retrieval. *ICCV*.

[Sun et al. 2018a] Sun, B.; Shao, M.; Xia, S.-Y.; and Fu, Y. 2018a. Deep evolutionary 3d diffusion heat maps for large-pose face alignment. In *BMVC*.

[Sun et al. 2018b] Sun, K.; Li, M.; Liu, D.; and Wang, J. 2018b. Igcv3: Interleaved low-rank group convolutions for efficient deep neural networks.

[Sun, Wang, and Tang 2013] Sun, Y.; Wang, X.; and Tang, X. 2013. Deep convolutional network cascade for facial point detection. In *CVPR*, 3476–3483.

[Szegedy et al. 2015] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *CVPR*, 1–9.

[Szegedy et al. 2017] Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A. A. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, 12.

[Tan et al. 2018] Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; and Le, Q. V. 2018. Mnasnet: Platform-aware neural architecture search for mobile. *arXiv preprint arXiv:1807.11626*.

[Trigeorgis et al. 2016] Trigeorgis, G.; Snape, P.; Nicolaou, M. A.; Antonakos, E.; and Zafeiriou, S. 2016. Mnemonic descent method: A recurrent process applied for end-to-end face alignment. In *CVPR*, 4177–4187.

[Wen et al. 2016] Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2016. Learning structured sparsity in deep neural networks. In *NIPS*, 2074–2082.

[Xiong and De la Torre 2013] Xiong, X., and De la Torre, F. 2013. Supervised descent method and its applications to face alignment. In *CVPR*, 532–539.

[Yang et al. 2018] Yang, T.-J.; Howard, A.; Chen, B.; Zhang, X.; Go, A.; Sandler, M.; Sze, V.; and Adam, H. 2018. Netadapt: Platform-aware neural network adaptation for mobile applications. *ECCV* 41:46.

[Yang, Liu, and Zhang 2017] Yang, J.; Liu, Q.; and Zhang, K. 2017. Stacked hourglass network for robust facial landmark localisation. In *CVPR Workshop*, 2025–2033.

[Yuan and Lin 2006] Yuan, M., and Lin, Y. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68(1):49–67.

[Zhang et al. 2018] Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*.

[Zhu et al. 2016] Zhu, X.; Lei, Z.; Liu, X.; Shi, H.; and Li, S. Z. 2016. Face alignment across large poses: A 3d solution. In *CVPR*, 146–155.