Supplementary Material

Anonymous ECCV submission

Paper ID 777

1 Convolutional Sensing Module (CSM)

CSM, shown in Fig. 1, senses the original image by gradually compact the image size through a sequence of filters. Compared with conventional sensing strategies, which sense the image block by block through a single shared weight matrix multiplication, this new approach has the power to preserve the spatial features thanks to the sparse local connectivity nature of convolution operations.

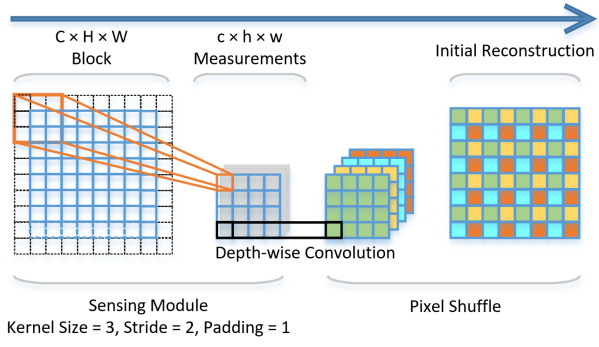

Fig. 1. Convolutional Sensing Module

The number of feature channels of intermediate layers of CSM during training can be relatively large, as long as the final output shape can meet the required measurements. Since there are no activation functions and no biases, no matter how wide the CSM is, these linear combinations can be finally squeezed into one matrix multiplication as shown in Fig. 2.

To be more specific, we take the feed forward network as an example. Assume that the network input is an n_0-elements vector x, the l^{th} layer contains n_l hidden cells and the i^{th} hidden cell in l^{th} layer is denoted as $h_{l,i}$, the weight in l^{th} layer connecting $h_{l-1,i}$ and $h_{l,j}$ is $w_{l,i,j}$. Then the feed forward network can be modeled

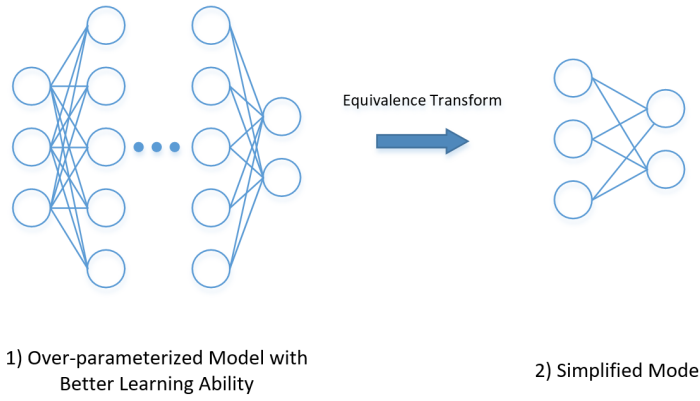

Fig. 2. Simplifying Linear Over-parameterized Model

by:

$$
\begin{aligned}
h_{0,i} &= \sum_{j=0}^{n_0} w_{0,i,j} x_j \\
h_{1,i} &= \sum_{j=0}^{n_1} w_{1,i,j} h_{0,j} \\
h_{2,i} &= \sum_{j=0}^{n_2} w_{2,i,j} h_{1,j} \\
&\vdots \\
h_{l,i} &= \sum_{j=0}^{n_l} w_{l,i,j} h_{l,j}
\end{aligned}
$$

Every subsequent hidden cell can be represented as a linear combination of x as follows:

$$
\begin{aligned}
h_{l,i} &= \sum_{j_l=0}^{n_l} w_{l,i,j} \sum_{j_{l-1}=0}^{n_{l-1}} w_{l-1,i,j} \cdots \sum_{j_0=0}^{n_0} w_{0,i,j} x_j \\
&= \sum_{j_l=0}^{n_l} \sum_{j_{l-1}=0}^{n_{l-1}} \cdots \sum_{j_0=0}^{n_0} w_{0,i,j} w_{1,i,j} \ldots w_{l,i,j} x_j \\
&= \sum_{j=0}^{n_0} W_{i,j} x_j
\end{aligned}
$$

This indicates that the final output of the network y can also be represented as a linear combination of the input x. Thus we can utilize the learning ability of an over-parameterized model to converge to a better optimal point. However, wider model is more likely to cause unstable gradient problems during training. So it is a trade-off to choose a proper width.

2 Runge-Kutta Block (RKB)

As the gradual reconstruction of the image can be reformulated as the progression of a dynamical system, we thereby apply Runge-Kutta methods, which is a family of high precision single step algorithms for numerical solutions of Ordinary Differential Equations (ODE), to build a novel residual architecture with better performance than conventional skip connection of ResNet. Specifically, Runge-Kutta methods take the form:

$$
\begin{aligned}
y_{n+1} &= y_n + \sum_{i=1}^{s} b_i K_i \\
K_1 &= hF(x_n, y_n) \\
K_2 &= hF(x_n + c_2 h, y_n + h(a_{21} K_1)) \\
K_3 &= hF(x_n + c_3 h, y_n + h(a_{31} K_1 + a_{32} K_2) \\
&\vdots \\
K_i &= hF(x_n + c_i h, y_n + h \sum_{j=1}^{s} a_{ij} K_j)
\end{aligned}
$$

where s denotes the number of stages to be specified, a_{ij}, b_i and c_i are the coefficients, K_i is the tangent slope at $x_n + c_i h$. When it comes to the second-order case, it becomes:

$$
\begin{aligned}
y_{n+1} &= y_n + b_1 K_1 + b_2 K_2 \\
K_1 &= hF(x_n, y_n) \\
K_2 &= hF(x_n + c_2 h, y_n + a_{21} K_1)
\end{aligned}
$$

To specify the exact values of the coefficients, we expand K_2 at (x_n, y_n), and according to Taylor's formula:

$$
\begin{aligned}
& hF(x_n + c_2 h, y_n + a_{21} K_1) \\
= \;& h\left[F + c_2 h {F'}_x + a_{21} K_1 {F'}_y + O(h^2)\right] \\
= \;& h\left[F + c_2 h {F'}_x + a_{21} h F {F'}_y + O(h^3)\right]
\end{aligned}
$$

where F denotes $F(x_n, y_n)$ and F_x, F_y are the partial derivatives of F with respect to x and y, respectively. Then we get:

$$
\begin{aligned}
y_{n+1} &= y_n + (b_1 K_1 + b_2 K_2) \\
&= y(x_n) + b_1 hF + b_2 h \left[F + c_2 h {F'}_x + a_{21} h F {F'}_y + O(h^3) \right] \\
&= y(x_n) + (b_1 + b_2) hF(x_n, y_n) + c_2 b_2 h^2 {F'}_x + a_{21} b_2 h^2 F {F'}_y + O(h^3)
\end{aligned}
$$

And Taylor's expansion of $y(x_{n+1})$ at x_n is:

$$
\begin{aligned}
y(x_{n+1}) &= y(x_n) + hy'(x_n) + \frac{h^2}{2!} y''(x_n + O(h^3) \\
&= y(x_n) + hF(x_n, y_n) + \frac{h^2}{2!} \left[{F'}_x + F {F'}_y \right] + O(h^3)
\end{aligned}
$$

Let $y(x_{n+1}) = y_{n+1}$, we get:

$$
\begin{aligned}
b_2 + b_1 &= 1 \\
b_2 c_2 &= \frac{1}{2} \\
b_2 a_{21} &= \frac{1}{2}
\end{aligned}
$$

which is an underdetermined system of equation, all methods satisfying the above forms are collectively referred to as the Second-Order Runge-Kutta Method. As we can see, b_2 is the only free variable that can be jointly optimized during training, and the final expression of the whole process is:

$$
y_{n+1} = y_n + [(1 - b_2) K_1 + b_2 K_2]
$$

$$
K_1 = hF(x_n, y_n)
$$

$$
K_2 = hF(x_n + \frac{h}{2b_2}, y_n + \frac{1}{2b_2} K_1)
$$

It can be seen that the standard format of Second-Order Runge-Kutta Method has only one free variable of b_2.