卷积神经网络对抗攻击与防御实验

黄 x 19000xxxxx
北京大学生命科学学院
生物信息学专业

日期：2022 年 5 月 6 日

1 引言

在计算机视觉领域中，CIFAR-10 是重要的一个数据集，也是作为机器学习中一个公开的基准测试数据集。该数据集共由 60000 张 32×32 的 RGB 三通道彩色图片组成，其中 10 个类别分别为：飞机（plane）、汽车（car）、鸟（bird）、猫（cat）、鹿（deer）、狗（dog）、青蛙（frog）、马（horse）、船（ship）、卡车（truck），并且每个类别各有 6000 张图片。我们以每个类各取 5000 张图片作为训练数据，其余 1000 张图片作为测试数据，分别用于训练、测试我们的模型，作为模型分类准确率的评估标准。

这里我们参考了 AlexNet 与 LeNet-5 的设计 [1-2]，搭建了一个如图 1 所示的简单的卷积神经网络，其中每个卷积层形如 Conv-BN-ReLu-AvgPool，全连接层则为 Linear-ReLu。在训练过程中，我们使用 CrossEntropy 作为损失函数，用 SGD 作为优化方法，学习率固定为 0.001，每个训练步骤使用 100 个训练集中的样本，并进行共计 10 次迭代，以此得到我们的模型。

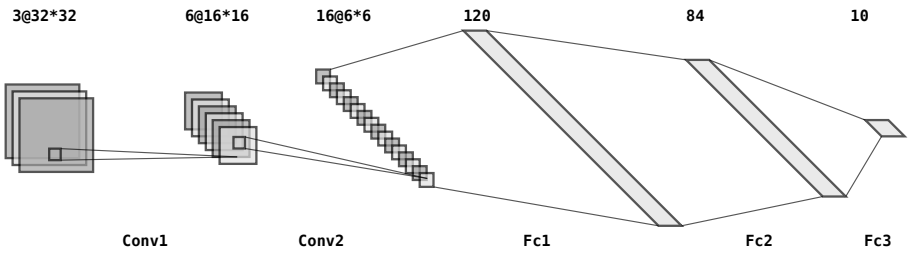

图 1: 卷积网络模型示意

2 对抗训练

尽管如此相对简单的神经网络能够在分类任务上取得相对不错的准确率，但针对如此已知结构的神经网络，在此工作 [3] 中，作者提出了一种基于梯度的攻击方法，即映射式梯度下降（Project Gradient Descent, PGD），算法框架如1。

算法 1: 映射式梯度下降攻击

Data: Max attack amplitude ϵ, Attack step size α, Iteration number N

Input: Model $M(\cdot)$, Original image x, Ground truth label y, Loss function $\mathcal{L}(\cdot)$

Result: Attack image x_N

Initialization: $x_0 = x + \xi$, where $\xi \sim \mathcal{N}(0, \epsilon)$;

for $i \leftarrow 1$ **to** N **do**

  $x_i = x_{i-1} + \alpha \cdot \operatorname{sgn}\left(\nabla_{x_{i-1}} \mathcal{L}\left(x_{i-1}, y \mid M\right)\right)$;

  $x_i \leftarrow \min\left(\max\left(x - \epsilon, x_i\right), x + \epsilon\right)$;

  `// clamp to` $\sim 95\%$ `CI of` $\mathcal{N}(0,1)$`, can be omitted`

  $x_i \leftarrow \min\left(\max\left(-2, x_i\right), 2\right)$;

end

return x_N

使用算法1，我们能够得到针对特定模型的攻击图像，尽管其与原图像的差异较小（如图2），但却能使得模型产生误判，得到完全不同的分类结果，因而达到攻击目的。在原工作中，Madry 等人也证明了这种攻击方式是仅采用网络中局部一阶信息（即梯度）所能实现的最强的攻击方式 [3]。因此我们的网络如果能够很好地应对 PGD 攻击，则其理应具有抵抗其他类型攻击的能力。

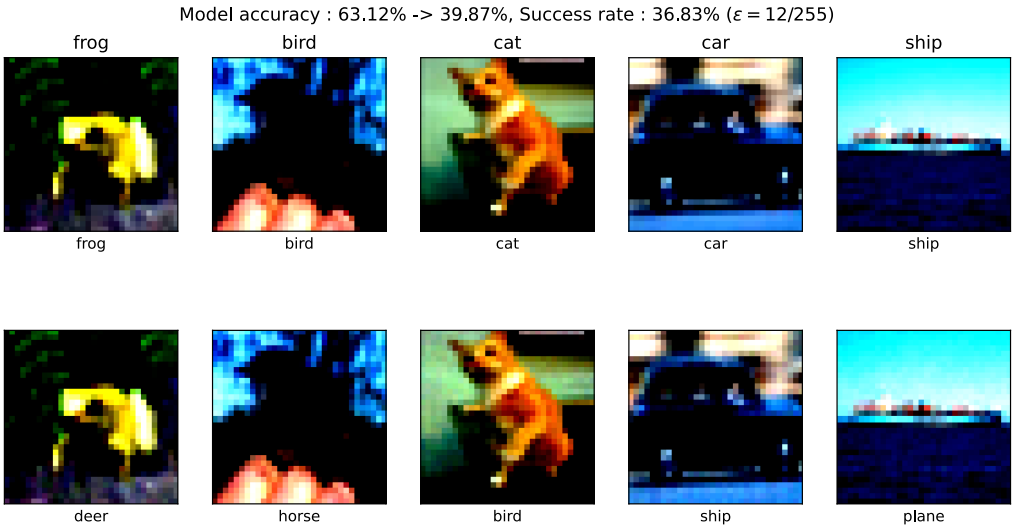

图 2: PGD 攻击前后的图像

我们利用上述结论，在训练过程中仅使用 PGD 攻击生成的对抗样本作为输入来训练网络，以此确保网络本身的鲁棒性，结果见第6部分。

3 模型量化

除去直接使用对抗样本进行训练，我们可以从对模型的参数做出一些调整与限制，提高其泛化能力，以求得更好的防御效果。其中一种方法为对模型进行量化，限制模型的精度，使其获得对于小扰动不敏感的能力，因而提高其泛化能力 [4]。对于模型的量化大致可分为两类 [5]:

一种模式为将所有涉及的参数与数据转化为整型，因此可使得矩阵计算更高效并且减小模型规模，但是通常并不能保证提供更高的精度或者更好的泛化能力（图3左）；另一种模式为仅对网络中权重和/或激活函数进行量化，尽管数据类型仍为浮点数，但是降低了数据的位深度，进而降低了模型的复杂度（图3右）。

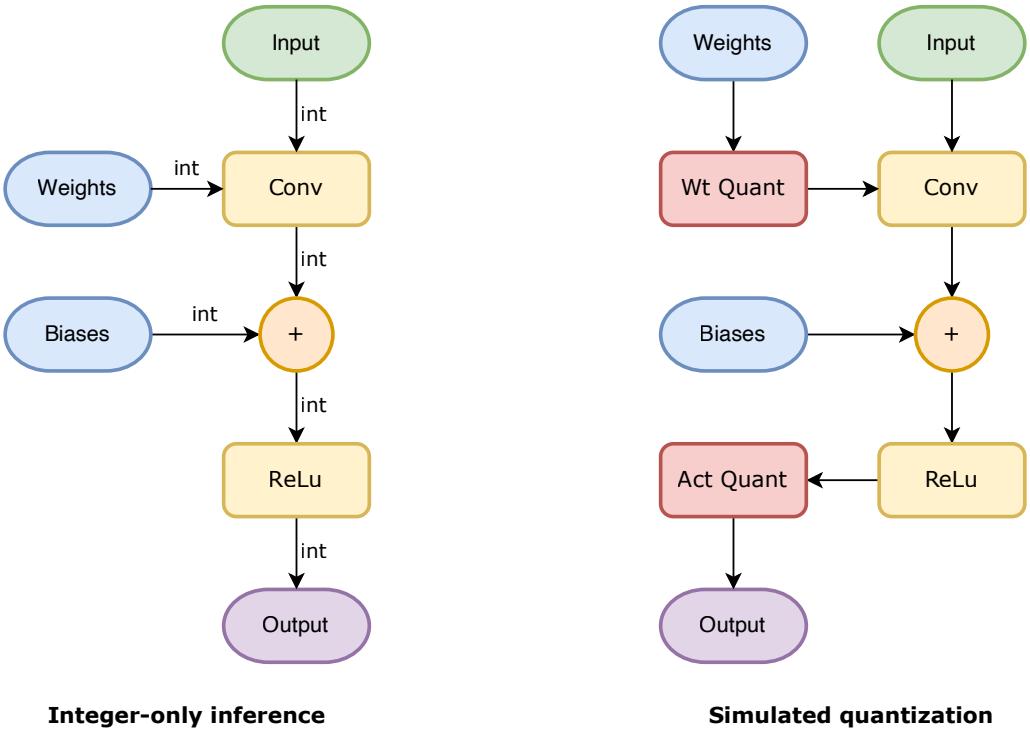

图 3: 两种量化模型对比

左：整型网络（仅推断），右：模拟量化（量化感知训练）

此处我们的目标是运用模型量化提高泛化能力，因此我们仅使用模拟量化的方式，对模型进行训练。对于一个模型而言，我们希望量化能提供泛化能力，在数学上即为对于输入值的不敏感，因此模型中仅有全连接层 `fc`、卷积层 `conv` 与我们此处选用的激活函数 `relu`能实现此目的。其中对于全连接层和与卷积层等效的全连接层而言，对于任意小扰动应有

$$\begin{aligned} \|f(x+\Delta x)-f(x)\|_2 &= \|(W(x+\Delta x)+b)-(Wx+b)\|_2 \\ &= \|W\Delta x\|_2 \end{aligned} \tag{1}$$

3.1 权重量化

由（1）式可得，仅有权重矩阵与扰动产生的效果相关，因而我们仅需处理其权重矩阵，对于其偏置等其他参数可不做特殊处理。对于权重矩阵的量化，我们可采用如下算法进行模拟。

算法 2: 模拟量化

Data: Lower bound a, Upper bound b, Bits resolution N

Input: Input x

Result: Quantized output x_{quant}

Initialization: **if** not given a or b **then**

  $a \leftarrow \min x$;

  $b \leftarrow \max x$;

end

Scale $s \leftarrow \frac{b-a}{2^N-1}$;

$x_{int} \leftarrow \left\lfloor \frac{\min(\max(a,x),b)-a}{s} \right\rceil$;

$x_{quant} \leftarrow s \cdot x_{int} + a$;

return x_{quant}

算法2中使用了 `round` 函数，因而此处的梯度会被置为 0，故我们需要对其进行一些补偿使得网络得以训练，这里我们参考饱和 Straight-through Estimator 方式 [6]，将超出上下界部分的梯度置为 0，其余部分保持不变，见 (2) 式。

$$\begin{aligned}\frac{\partial f(x_{quant})}{\partial x} &= \frac{\partial f(x_{quant})}{\partial x_{quant}} \circ 1_{x\in[a,b]} \\ &= \begin{cases} \frac{\partial f(x_{quant})}{\partial x_{quant}} & \text{if } x \in [a,b] \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

where $\circ :=$ Element-wise product

3.2 量化激活函数

尽管如算法2所示的算法直接应用于权重时通常效果较好，但是在应用于可能遇到极值的激活函数时则可能会遇到问题，因此这里我们参考了一种较为简单但能够处理离群值的方法 [5]，即指定量化的上下界 a, b，在训练过程中使用指数滑动平均来减弱离群值带来的影响，同时能够在推断过程中对离群值进行限制。这里我们采用一种等效的形式，即为上下界赋予梯度，使其缓慢学习该层的输入值。更进一步的，我们还可以采用全部动态阈值阈值的激活方式，使每个单独的阈值可学习，附录图11中包含了不同的一些实现。不难发现，仅有 `DTReLu`（动态上下界）与 `DDTReLu`（动态阈值）两种函数与我们的预期近似，结合图12中展示的运算复杂度，我们此处仅选用 `DTReLu` 作为可学习的阈值激活函数。

4 正则化约束

为了实现网络对于扰动的不敏感性，我们也可以从限制网络李普希茨常数的角度出发，使得网络收敛到相对平滑的位置，避免陷入局部最优，因此也可以一定程度上给予更佳的防御能力。

4.1 正交正则化

卷积与线性层的操作可以看做是将输入进行一定的线性变换至新的空间，因此如果我们能够使线性变换中所用的基相对正交，则可以避免变换后参数空间出现退化的可能。以线性层为例，结合 (1) 式，我们只需要对其权重矩阵引入正交化激励，即可达到此效果。这里我们总结了常见的多种正交正则化项的实现 [7-8]，见 (3) 式，经过测试不同的激励项之间效果差异不大，因此我们选择最后一种实现，以期望在权重矩阵较小时仍然能够达到正交化的效果。

$$\mathcal{L}(W) = \begin{cases} \left\| W^T W - I \right\|_2 \\ \left\| W W^T - I \right\|_2 \\ \left\| W^T W \circ (1_W - I) \right\|_2 \\ \left\| \frac{W^T W}{\|W^T W\|_2} - I \right\|_2 \end{cases} \tag{3}$$

这里由于卷积层的权重矩阵为一四维的矩阵，形如 (4) 式，因此我们需要对其做出处理使其能够参与计算。

$$W_{Conv2d} : [C_{out}, C_{in}, K, K]$$
$$\text{where} \begin{cases} C_{in}: & \text{输入通道数} \\ C_{out}: & \text{输出通道数} \\ K: & \text{卷积核高宽} \end{cases} \tag{4}$$

考虑到每个卷积操作的输出高宽均为 1，因此如图4所示，我们可以将其后三维展平，得到一个二维矩阵，即可按照线性层权重矩阵的方式进行计算。

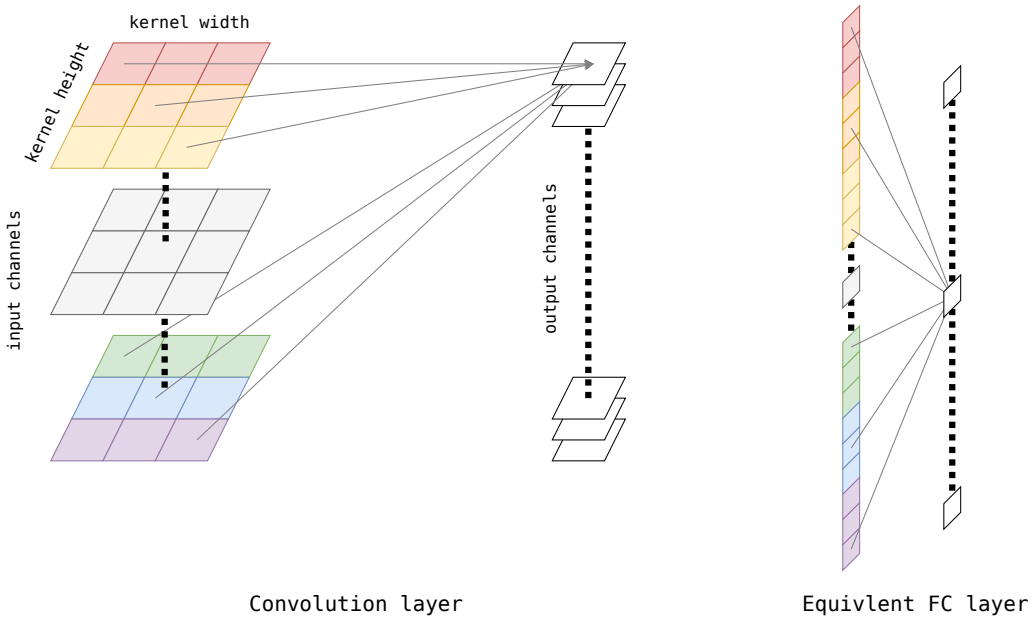

图 4: 卷积层等效线性层简单实现

左：卷积核示意图，右：卷积核等效的线性层

4.2 谱范数正则化

考虑网络中每个卷积层或线性层操作，继续 (1) 式有：

$$\begin{aligned}\|f(x+\Delta x)-f(x)\|_2 &= \|W\Delta x\|_2 \\ &\leq \|W\|_2 \|\Delta x\|_2 \\ &= \max\{\sigma(W)\}\cdot\|\Delta x\|_2 \\ &= \sigma_0(W)\cdot\|\Delta x\|_2\end{aligned} \tag{5}$$

其中 $\sigma_0(W)$ 即为矩阵的最大奇异值，我们可以采用幂迭代法快速逼近此式，对与线性层的二维权重矩阵，我们有如算法3的实现。

算法 3: 幂迭代法求最大奇异值（线性层）

Data: Number of iterations N

Input: Weight matrix $W \in \mathbb{R}^{h\times w}$

Result: First singular value σ

Get height h and weight w of W;
$u \leftarrow \mathbf{1} \in \mathbb{R}^h$ and $v \leftarrow \mathbf{1} \in \mathbb{R}^w$;
for $i \leftarrow 1$ **to** N **do**
    $u' \leftarrow \frac{Wv}{\|Wv\|_2}$;
    $v' \leftarrow \frac{W^T u}{\|W^T u\|_2}$;
    $u \leftarrow u'$;
    $v \leftarrow v'$;
end
$\sigma \leftarrow u^T W v$;
return σ

而对于卷积层，我们可以参照原文章的做法 [9]，对卷积层权重做如图4的变换后使用上述算法3进行处理。然而考虑到卷积层实际的效果，卷积核中的每个权重值对于实际操作的贡献程度有出入（取决于其位置，如越靠近中间的权重参与了更多的运算），因此我们实际需要考虑一个等效的稀疏操作矩阵，如图5所示。若我们的卷积核仍如 (4) 式所示，则对应的稀疏权重矩阵形如 (6) 式（仅考虑步长为 1）。对于图5左侧这样的卷积操作，我们正则化约束的目标现在即为图5右侧所示的矩阵。

$$\begin{aligned}&W_{EquivLinear}:[C_{out}\times H_{out}\times W_{out}, C_{in}\times H_{in}\times W_{in}] \\ &\text{where}\begin{cases} H_{in}: & \text{输入高度} \\ W_{in}: & \text{输入宽度} \\ H_{out}=H_{in}-K+2\times P+1: & \text{输出高度} \\ W_{out}=W_{in}-K+2\times P+1: & \text{输出宽度} \\ P: & \text{填充}\end{cases}\end{aligned} \tag{6}$$

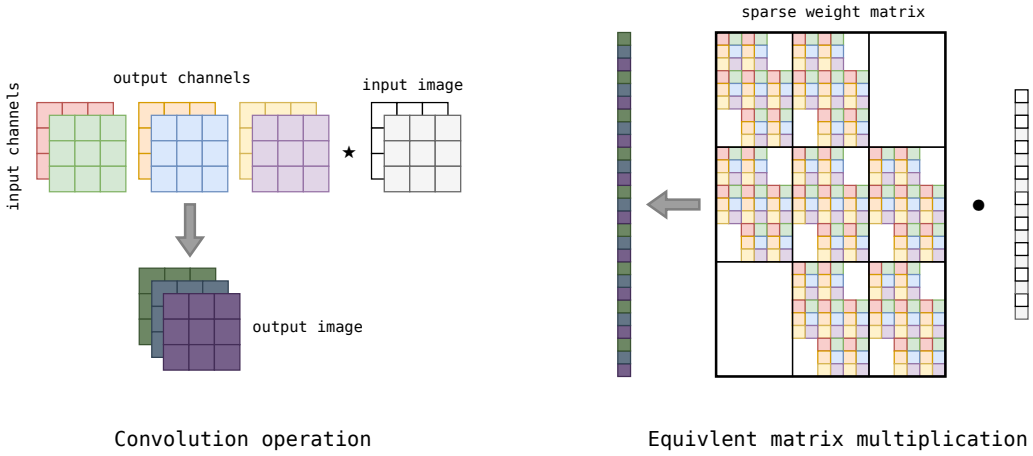

图 5: 卷积等效线性映射

左：卷积操作示意图，右：卷积操作等效矩阵乘法

我们观察图5所示的稀疏矩阵及其转置，则可反推得到稀疏矩阵转置对应的卷积核，即有：

$$\begin{aligned} &W_{EquivLinear}^{T} : [C_{in} \times H_{in} \times W_{in}, C_{out} \times H_{out} \times W_{out}] \\ &\Rightarrow W_{Conv2d}^{Reverse} : [C_{in}, C_{out}, -K, -K] \\ &\text{and } P^{Reverse} = K - 1 - P \\ &\qquad \text{where } -K : \text{沿原轴翻转} \end{aligned} \tag{7}$$

因此我们对于算法3略作修改，可得到以下适用于卷积操作的幂迭代算法4，避免实际上转化为稀疏矩阵，因而避免浪费内存与计算性能。

算法 4: 幂迭代法求最大奇异值（卷积层）

Data: Number of iterations N, Input shape (C_i, H_i, W_i), Output shape (C_o, H_o, W_o), Kernel size K, Padding P

Input: Convolution kernel matrix $W \in \mathbb{R}^{C_o \times C_i \times K \times K}$

Result: First singular value σ of corresponding sparse matrix

Notations: $W^{\mathcal{R}} := W$ in shape of $[C_i, C_o, -K, -K]$;

Notations: $V \star W :=$ Convolution on V with kernel W;

Notations: $U|_P :=$ Extend U in last two dimensions with P number of zeroes (padding);

$U \leftarrow \mathbf{1} \in \mathcal{R}^{C_o \times H_o \times W_o}$ and $V \leftarrow \mathbf{1} \in \mathcal{R}^{C_i \times H_i \times W_i}$;

for $i \leftarrow 1$ **to** N **do**

  $U' \leftarrow V|_P \star W$;

  $V' \leftarrow U|_{K-1-P} \star W^{\mathcal{R}}$;

  $U \leftarrow \frac{U'}{\|U'\|_2}$;

  $V \leftarrow \frac{V'}{\|V'\|_2}$;

end

$\sigma \leftarrow \sum U \circ (V|_P \star W)$;

return σ

5 模型集成

在分类器任务中，除了在模型层面作出改进提高分类性能，另一类重要的方法为集成。机器学习中的集成通常指采用不同的随机数种子，训练一批不同参数的模型实例，在预测时取所有实例的结果作无加权平均，以此作为最终的结果，通常预测的准确率超过单一的模型实例 [10]。基于此思路，后续又出现了自集成（Self-ensemble），其思想是在训练过程中对不同时刻的实例参数取平均，而非对最后的结果取平均，因而避免了在推断过程中高昂的计算成本，并且大大减小了模型大小 [11]。

5.1 随机自集成

在所述自集成之上更进一步，又出现了随机自集成（Random Self-ensemble, RSE）的方法 [12]，其做法与数据增广有一定可比之处，即对输入添加无意义的白噪声，但不同之处在于随机自集成的方法对于模型中每个如图6左所示的卷积模块之前对输入添加噪声，形成如图6右所示结构。在训练过程中加入噪声，使得其每一层的输入数据都得到增广，在不显著提升训练成本的同时，使得模型更容易收敛至平坦点而非极值点。而在推断过程中，同样加入噪声，可以达到模型增广的效果，得到“无限个”集成模型，因此也可提高模型的推断准确度，实现框架见算法5。原文章中实验结果为约在 10 个子集成模型时预测精度达到饱和 [12]，因此这里我们为了略微加速实验，只取 5 个子集成模型进行推断。

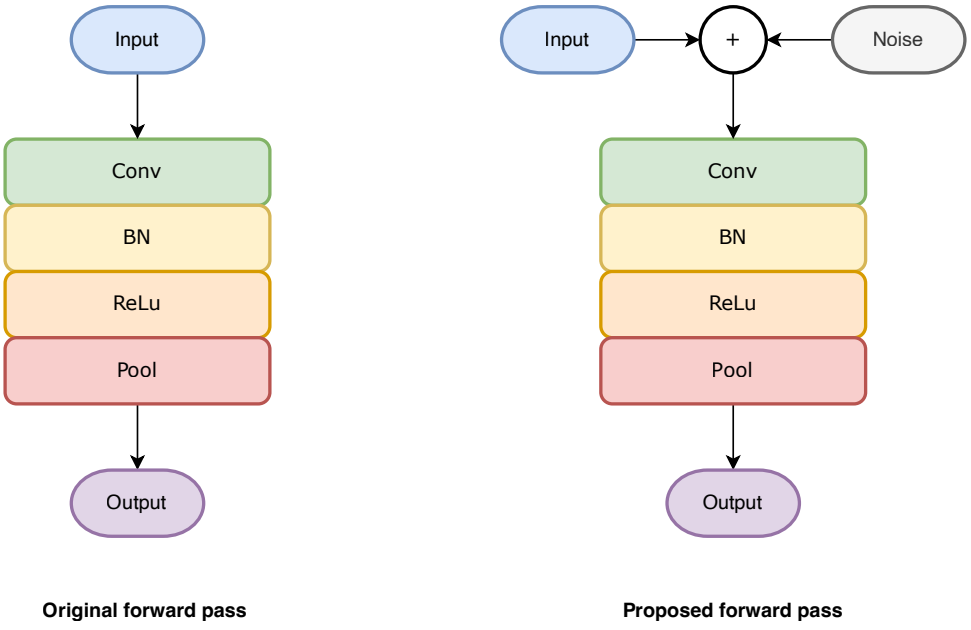

图 6: 随机自集成对模型的更改
（左）原始模型的卷积层结构，（右）随机自集成模型的卷积层结构

算法 5: 随机自集成模型

Data: Initial noise level σ_0, Inner layers' noise level σ_1 ($\sigma_1 < \sigma_0$), Number of ensembles N

Input: Model of k convolution layers $F = f_k\left(\cdots f_2\left(f_1(\cdot)\right)\cdots\right) =: f_1 \circ f_2 \circ \cdots \circ f_k$,
Input image x

Result: Predicted label y

if During training **then**
  `// Add extra noise before the first convolution layer`
  $x_1 \leftarrow x + \xi$ where $\xi \sim \mathcal{N}(0, \sigma_0)$;
  $y_1 \leftarrow f_1(x_1)$;
  for $l \leftarrow 2$ *to* k **do**
    $x_l \leftarrow y_{l-1} + \xi$ where $\xi \sim \mathcal{N}(0, \sigma_1)$;
    $y_l \leftarrow f_l(x_l)$;
  end
  $y \leftarrow y_k$;
else
  `// Average over all ensembles`
  $y \leftarrow \mathbf{0}$;
  for $i \leftarrow 1$ *to* N **do**
    `// Add extra noise before the first convolution layer`
    $x_1 \leftarrow x + \xi$ where $\xi \sim \mathcal{N}(0, \sigma_0)$;
    $y_1 \leftarrow f_1(x_1)$;
    for $l \leftarrow 2$ *to* k **do**
      $x_l \leftarrow y_{l-1} + \xi$ where $\xi \sim \mathcal{N}(0, \sigma_1)$;
      $y_l \leftarrow f_l(x_l)$;
    end
    $y \leftarrow y + \frac{1}{N} \cdot y_k$;
  end
end
return σ

由如上算法5我们同样可以知道，由于在推断时加入了多个模型的结果，因此在受到 PGD 攻击时，梯度会沿多条路径传播，因此其传回原始图片的多个梯度得到叠加，而在攻击的步长、幅值均有限时，这种叠加会由于其方向上的不同而产生削弱，因此也能较为显著地提高模型的防御能力。

6 结果

为了避免小概率事件导致的结果不准确，我们使用 10 个随机数种子来重复实验；并且对于每个模型，我们使用 4 个不同的 ϵ 值用于测试（和训练），因此共计 40 次重复实验。

6.1 训练效果对比

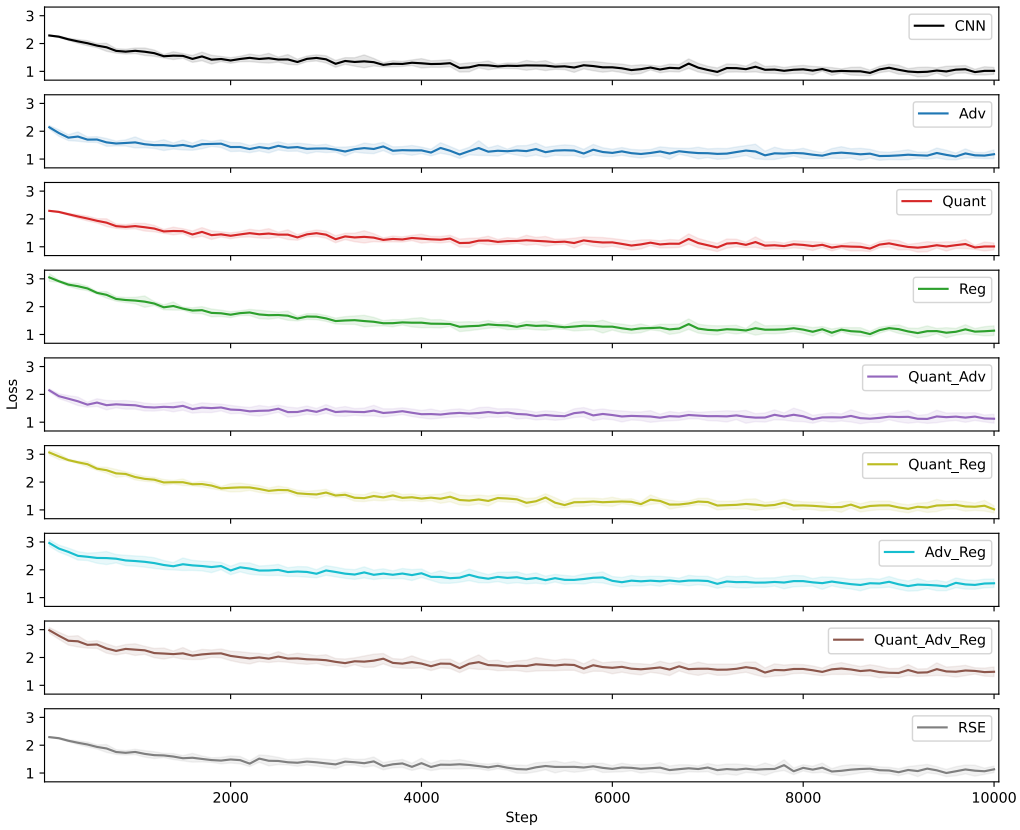

图 7: 不同模型训练过程中的损失

（黑）原始模型，（蓝）对抗训练，（红）量化模型，（绿）正则化约束，（紫）量化 + 对抗，（黄）量化 + 正则化，（青）对抗 + 正则化，（棕）量化 + 对抗 + 正则化，（灰）随机自组装

图7展示了不同模型在训练过程中损失函数值的变化，我们不难发现，与原始模型相比较，包含正则化项的模型（图7中绿黄青棕）在训练过程中收敛速度相对较慢，并且由于正则化项的存在一开始的损失函数值会较大；并且特别的有，当对抗训练与正则化约束相结合时（图7中青棕），在经过全部训练迭代次数后的损失函数值明显大于其他模型。其余模型在训练过程中的收敛速度相对接近，但其中不包含正则化项的对抗训练模型（图7中蓝紫）在初始几步内就迅速收敛，远快于其他修改后的模型与原始模型。

6.2 不同模型攻击成功率

我们对不同模型攻击前后的预测准确率进行统计，并且将改进后的模型结果与原始模型上的结果做单边成对 t-检验，即得到图8的结果。我们不难看出，所有带有对抗训练的模型其攻击前后的预测准确率均有显著提升，同时对其进行 PGD 攻击的成功率也有显著的下降。同时与图7结论一致的是，含有正则化项约束的模型在预测的准确率对应地略有下降。这也进一步说明正则化项在此处的作用与我们的设计初衷有所不同。除此之外，应对 PGD 攻击的防御能力得到显著提升的还有随机自组装的方法，尽管其在测试数据集上的预测能力并未得到显著提升，但对其进行 PGD 攻击的成功率却大大下降，与使用对抗训练的模型的攻击成功率较为接近。

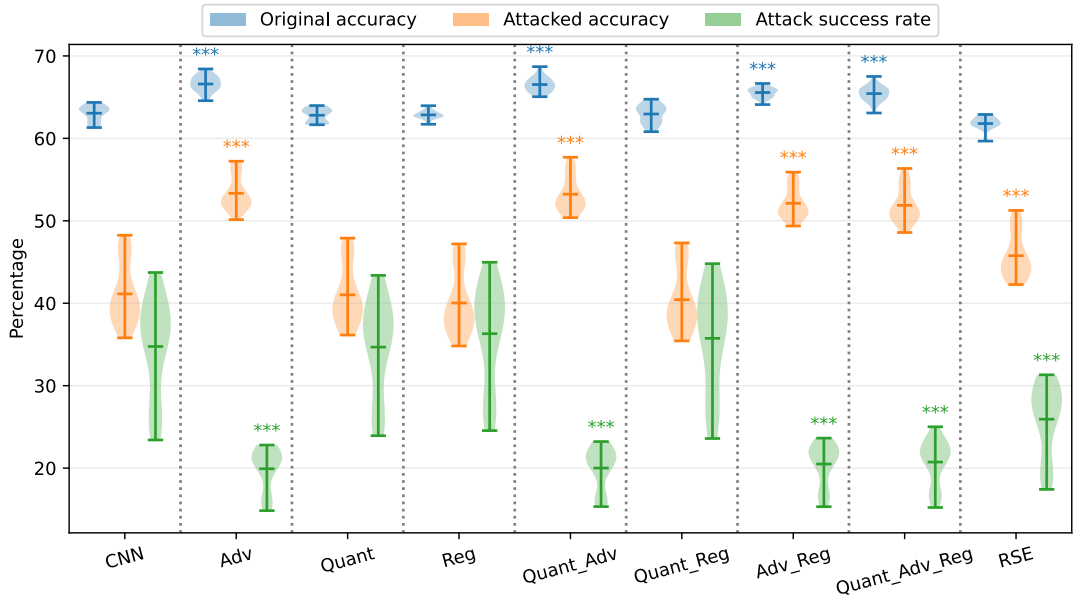

图 8: 不同模型攻击前后的预测准确率与 PGD 攻击成功率

（蓝）攻击前准确率，（橙）攻击后准确率，（绿）PGD 攻击成功率，

（左起依次为）原始模型，对抗训练，量化模型，正则化约束，量化 + 对抗，

量化 + 正则化，对抗 + 正则化，量化 + 对抗 + 正则化，随机自组装

***：$p < 0.01$，**：$p < 0.05$，*：$p < 0.1$

6.3 不同幅度的攻击成功率

在实验中我们测试了四种不同的 $\epsilon = 4/255, 8/255, 12/255, 16/255$，并且对于对抗训练的模型也使用同样水平的 ϵ 生成对抗样本，结果见图9，可以看出，由于我们仅变化了 ϵ，并未对 $\alpha = 2/255$ 和迭代数 $= 4$ 作出改变，因此随着 ϵ 的增大，攻击的成功率也逐渐饱和。与图8一致的是，包含有对抗训练的模型不管在 ϵ 取任何值时表现都优于其他方法或方法的组合。除此之外，我们发现随着 ϵ 的增大，PGD 攻击成功率的标准差都逐渐增大（图9中仅作示意，所展示的范围为实际标准差的对数值）。

7 讨论

7.1 正则化项约束与量化模型本身无法显著降低攻击成功率

在图8和图9的结果中，仅含有正则化和/或量化的模型并为能提升模型的防御能力。然而在如图7所示的损失函数变化中我们确实能见到正则化项的约束效果，使得网络收敛更为平滑。而仅使用量化的模型也确有相较于原始模型较小的精度损失。

对此我们认为原因主要在于我们使用的网络结构过浅，一是使得模型本身的表达能力有限，即便不加入正则化或量化，都不足以将输入数据作足够的非线性变换并映射到合适的位置，使得预测结果的区分度不大，故较难体现出正则化和量化的效果；二是正则化约束主要可以限制模型的平滑程度，然而在模型不发生过拟合、不陷入局部最优解时也较难体现效果，而采用了 STE 的量化方式也使得模型的变化较为平滑，因此同样只有在参数数量足够多、网络足够深时

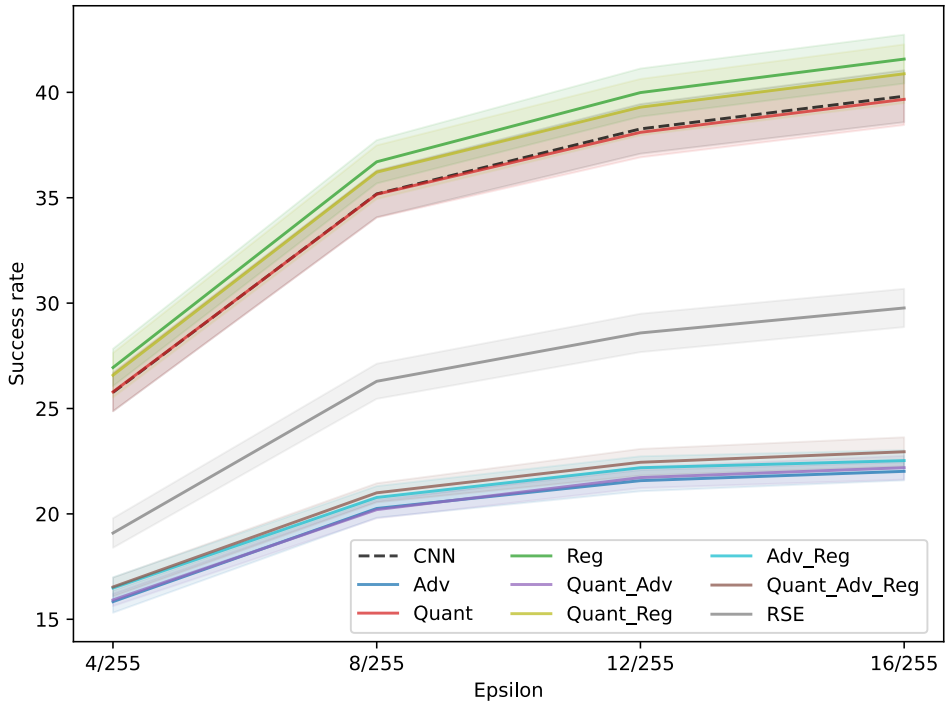

图 9: 攻击成功率随 ϵ 的变化

（黑）原始模型，（蓝）对抗训练，（红）量化模型，（绿）正则化约束，（紫）量化 + 对抗，（黄）量化 + 正则化，（青）对抗 + 正则化，（棕）量化 + 对抗 + 正则化，（灰）随机自组装

注：实线为真实均值，着色范围为对数变换后的标准差，仅作参考，不代表实际标准差

才能够体现出效果。

7.2 使用对抗训练的模型均得到显著提升

在比较所有结果后，不难认为使用对抗训练的模型防御效果近似，故分为一档；使用随机自集成的模型效果其次，独为一档；其余所有模型最次，又分为一档。更有意思的事，对抗训练不仅在防御任务上表现出色，在正常的、无对抗样本的测试集上的表现也全面超越了原模型，这在我们的意料之外。对此，我们的一种解释如图10，即 PGD 攻击主要使得经过神经网络变换的图像朝着分类边界移动，ϵ 实际限制的是移动的幅度。因此 PGD 攻击成功的部分主要为映射至分类边界处的数据，而当数据越靠近此边界或者 ϵ 越大时，攻击成功的样本就越多，即成功率越高。

在训练网络模型的过程中，往往不可避免的是当映射结果由错误的分类向正确的分类移动时，存在一条想象的包络线，即所有样本在朝正确方向移动时存在一定极限，越接近此极限，其在损失函数上的梯度就越不显著，从而对网络参数更新的贡献大幅降低。而在对抗训练的过程中，由于我们仅使用对抗样本进行训练，意味着每次模型都是沿着将分类边界的样本“推离”边界线的方向更新参数，因此使得训练过程中的参数更新非常的高效，相当于将原模型的包络线进一步向外推移并且对最坏情况作出了限制，故模型能够更好地区分原本落在分类边界线附近的数据点（图10下方），因而在预测数据集上也具有了更高的精度，同时这也解释了在图7中使用了对抗训练的模型都能够更快的收敛。

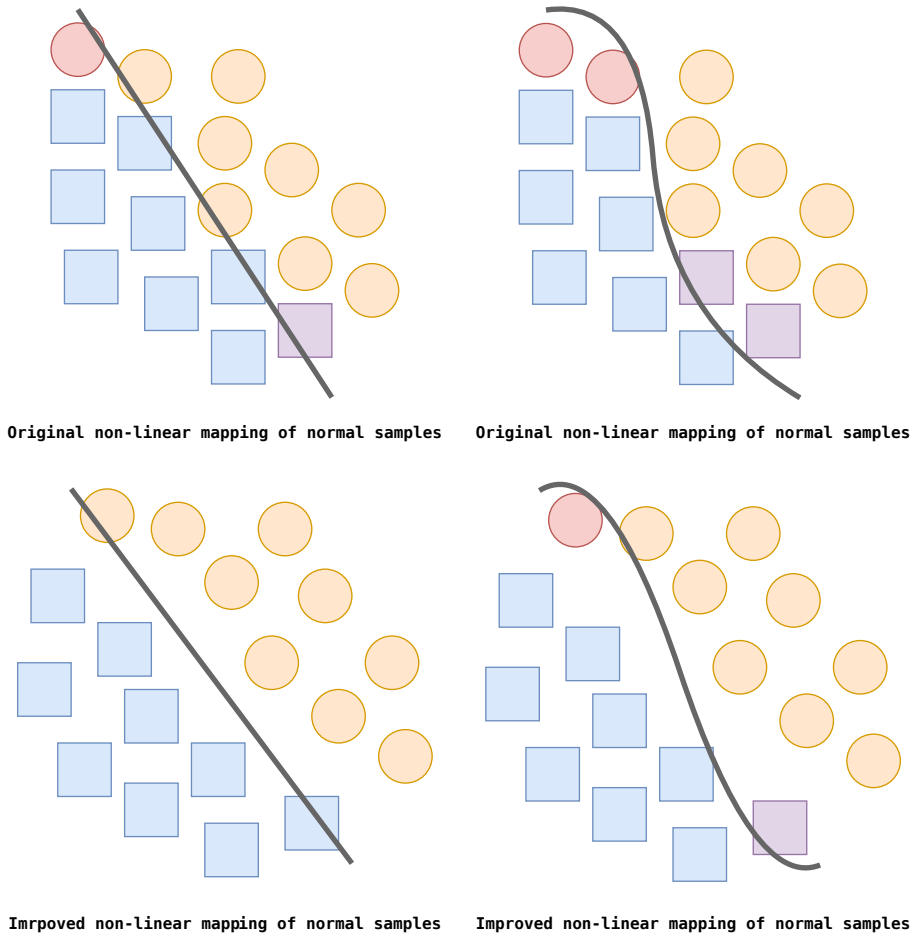

图 10: PGD 攻击与模型防御示意图

（上）未使用对抗训练，（下）使用对抗训练后，（左）正常测试集，（右）对抗样本测试集
灰线表示模型预测分类的边界，实际左右边界一致，仅有数据点发生移动，此处仅为示意

同理，我们也可以用于解释7.1中提出的问题，由于正则化约束与模型量化对于经过网络变换后的数据映射的作用是不定向的，因此我们实际上并不能保证其能够增加模型的防御能力。对比对抗训练的模式，我们可以认为使用对抗样本训练可以为网络中参数提供额外的梯度，并且几乎总是朝向令样本之间区分度变大的方向的，而正则化或量化并不能做到这一点，它们只能够将所有的数据点都进行“扩散”，并没有实质性改进模型分类的包络线，故在网络都收敛至相当程度时结果并未有所改善。

7.3 动态阈值激活函数选取问题

这里我们参考的是原文章中的实现 [5]，使用的是可变区间、等量化间距的激活函数，原因主要在于，一是全动态阈值的激活函数不易实现，在简单的实现中甚至可能是完全错误的，如图11蓝线实际上对输出值取了区间两端的平均，因此保证每个梯度都能够在其两边参与运算，使得自动求导的梯度能够正确回传，否则所有阈值的梯度仅有一个方向，会导致阈值漂移，进一步致使函数退化为二值函数，而仅依照阈值两端区间内元素个数来更新梯度（图11红）也会在某些情况下发生退化；二是此处函数的选取对于最终结果影响并不大，即便是采用二值函数网络都能够收敛，故我们选取了较为鲁棒的动态区间激活函数。

参考文献

1. Krizhevsky, A., Sutskever, I., and Hinton, G.E. (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM 60, 84–90.
2. Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324.
3. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks.
4. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W., and Keutzer, K. (2021). A survey of quantization methods for efficient neural network inference.
5. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. (2017). Quantization and training of neural networks for efficient integer-arithmetic-only inference.
6. Lin, X., Zhao, C., and Pan, W. (2017). Towards accurate binary convolutional neural network.
7. Brock, A., Lim, T., Ritchie, J.M., and Weston, N. (2016). Neural photo editing with introspective adversarial networks.
8. Bansal, N., Chen, X., and Wang, Z. (2018). Can we gain more from orthogonality regularizations in training deep CNNs?
9. Yoshida, Y., and Miyato, T. (2017). Spectral norm regularization for improving the generalizability of deep learning.
10. Rokach, L. (2009). Ensemble-based classifiers. Artificial Intelligence Review 33, 1–39.
11. Xu, Y., Qiu, X., Zhou, L., and Huang, X. (2020). Improving BERT fine-tuning via self-ensemble and self-distillation.
12. Liu, X., Cheng, M., Zhang, H., and Hsieh, C.-J. (2018). Towards robust neural networks via random self-ensemble. In Computer vision ECCV 2018 (Springer International Publishing), pp. 381–397.

附录

A 代码

A.1 网络模型

原始模型，对抗训练，量化模型，正则化约束，组合模型，随机自集成

A.2 杂项

超参数，PGD 攻击实现，公用函数，批量训练脚本，绘图脚本，动态阈值激活函数比较

B 附图

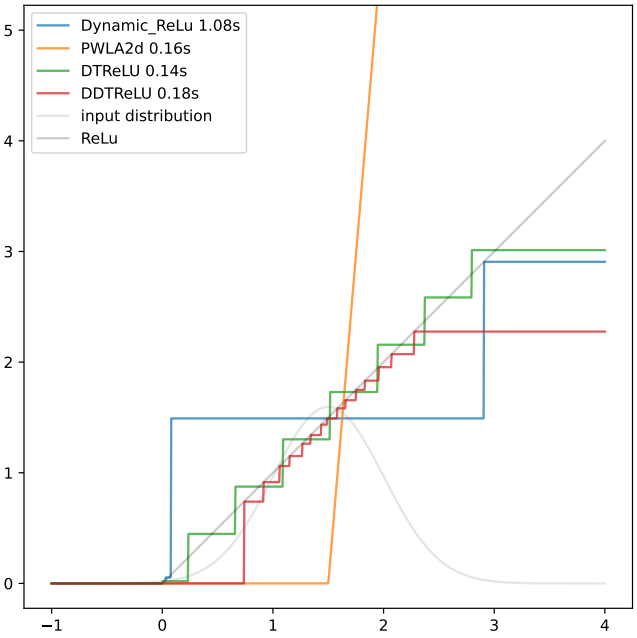

图 11: 动态阈值激活方式

（蓝）动态阈值激活函数的 naive 实现，（橙）分段线性函数，
（绿）动态区间激活函数，（红）动态阈值激活函数，（灰）训练数据分布，（深灰）ReLu 函数

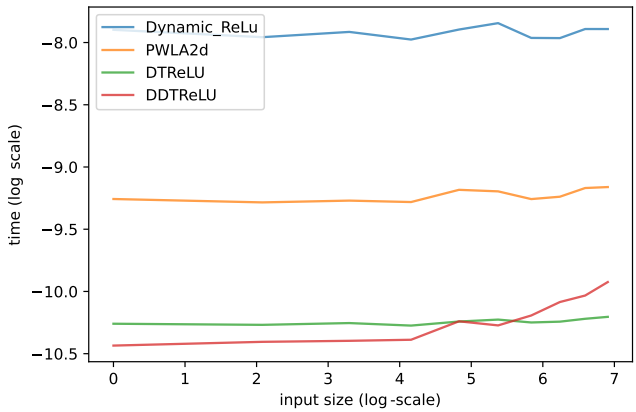

图 12: 不同动态阈值激活耗时随数据规模的变化

（蓝）动态阈值激活函数的 naive 实现，（橙）分段线性函数，
（绿）动态区间激活函数，（红）动态阈值激活函数

B.1 特征提取层可视化

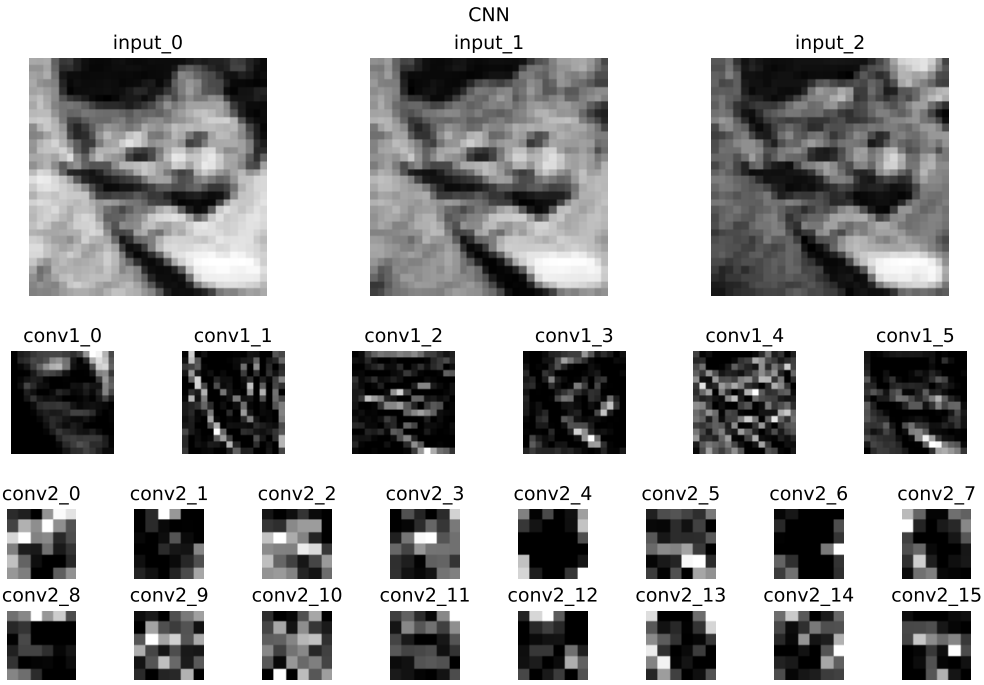

图 13: 原始模型特征图

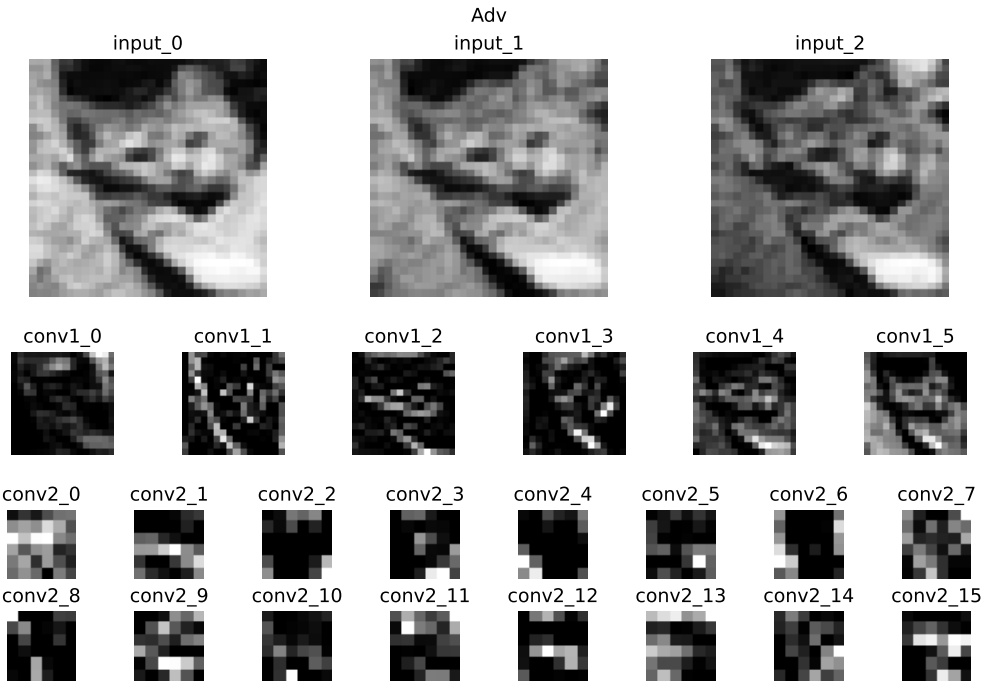

图 14: 对抗训练模型特征图

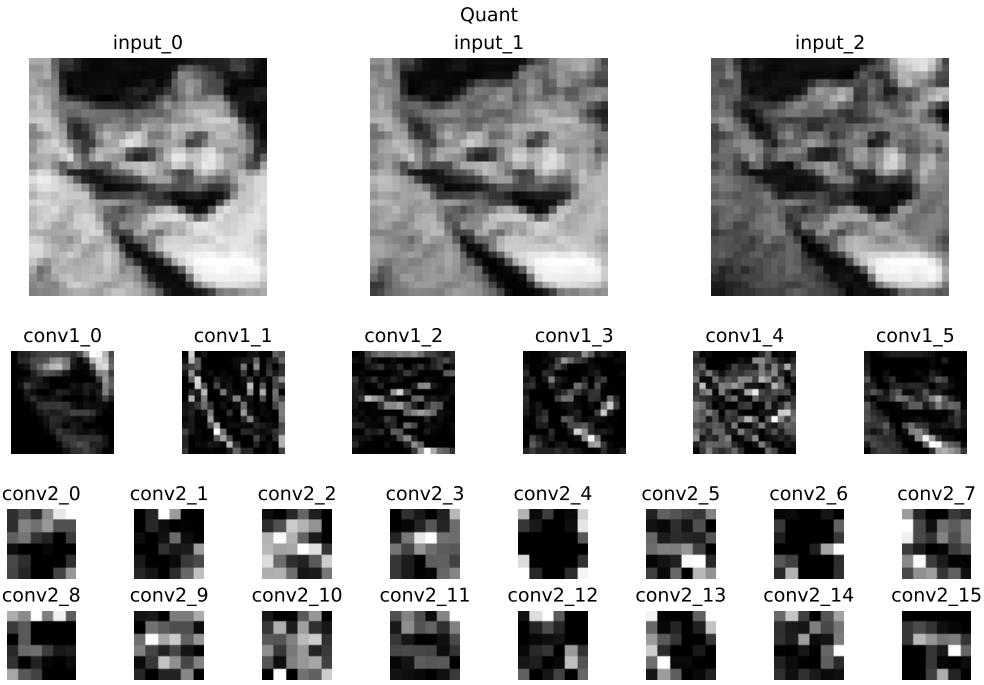

图 15: 量化模型特征图

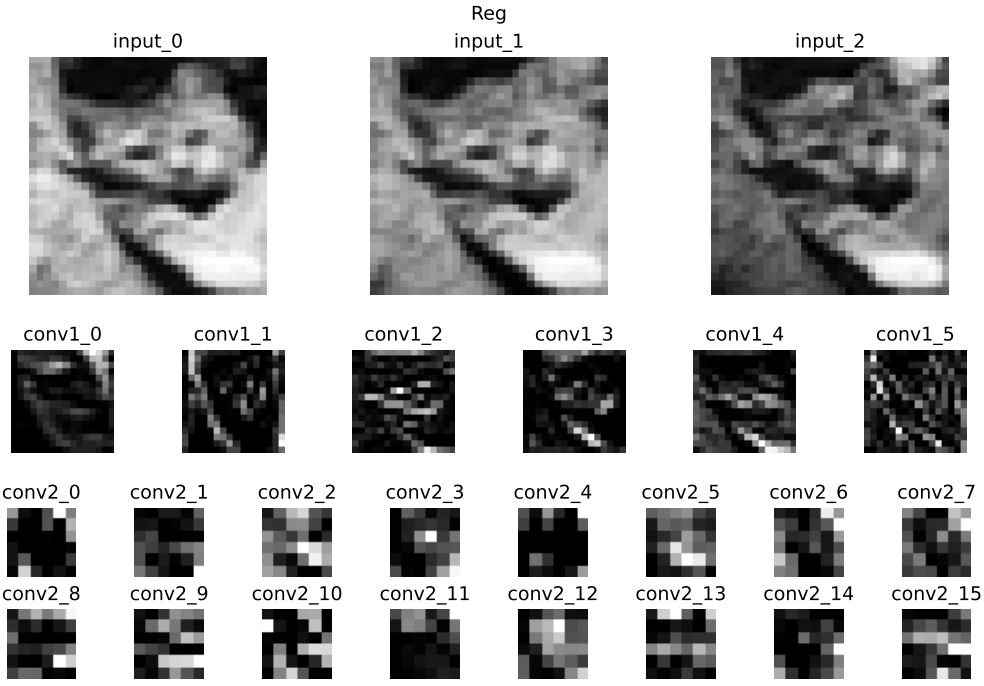

图 16: 正则化模型特征图

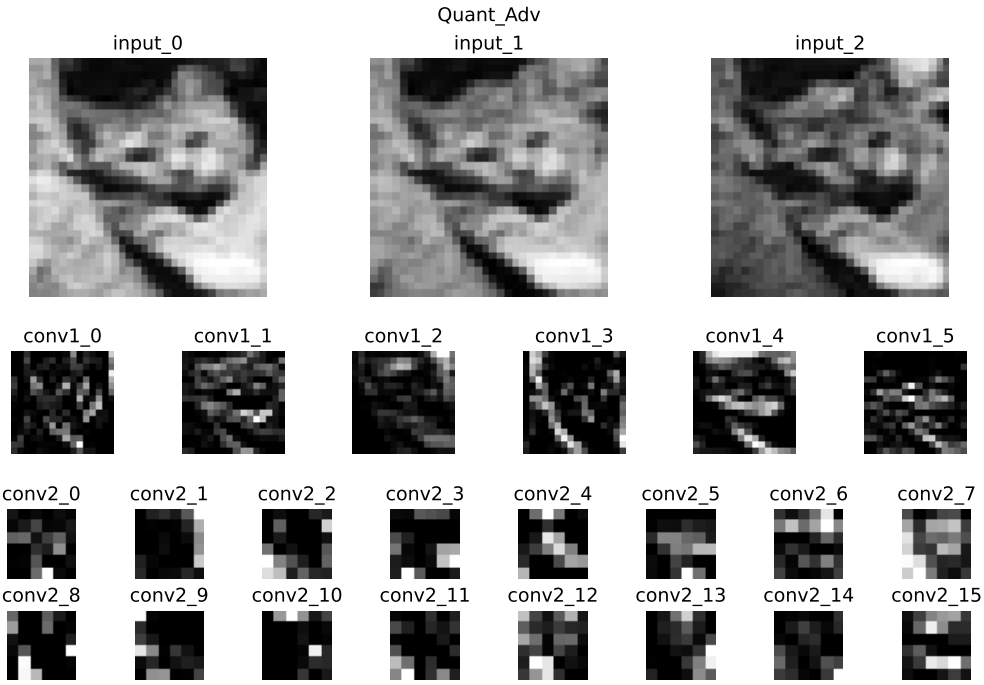

图 17: 量化 + 对抗训练模型特征图

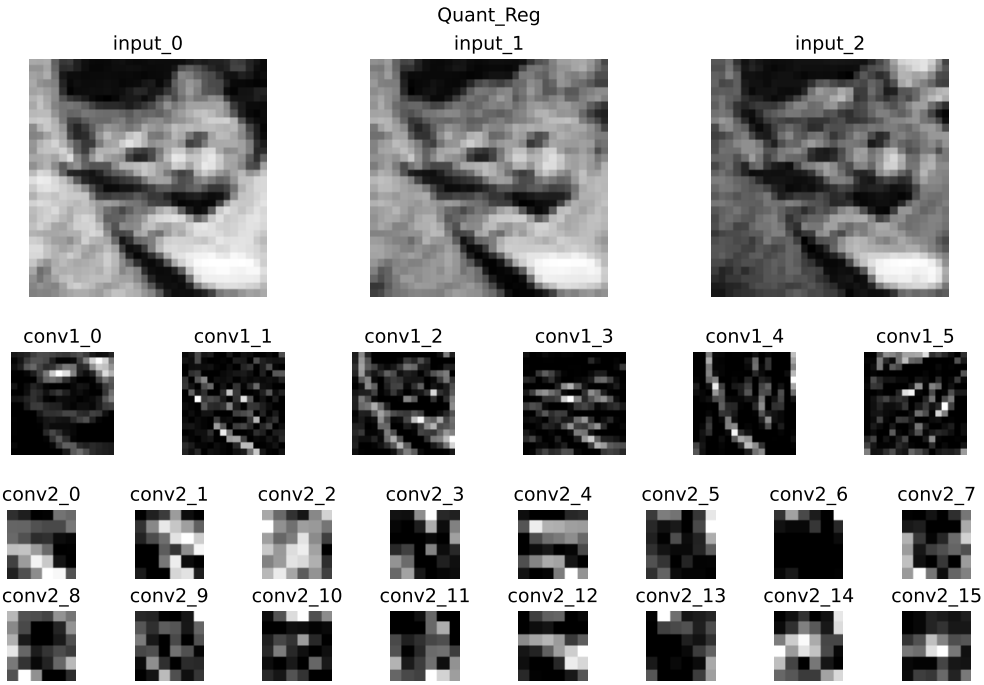

图 18: 量化 + 正则化模型特征图

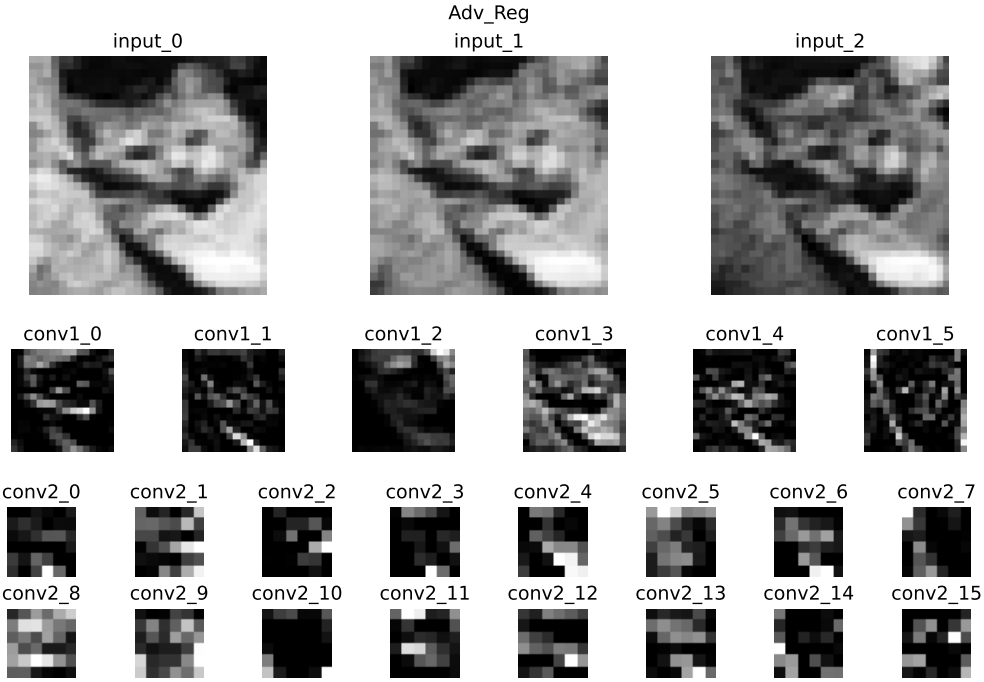

图 19: 对抗训练 + 正则化模型特征图

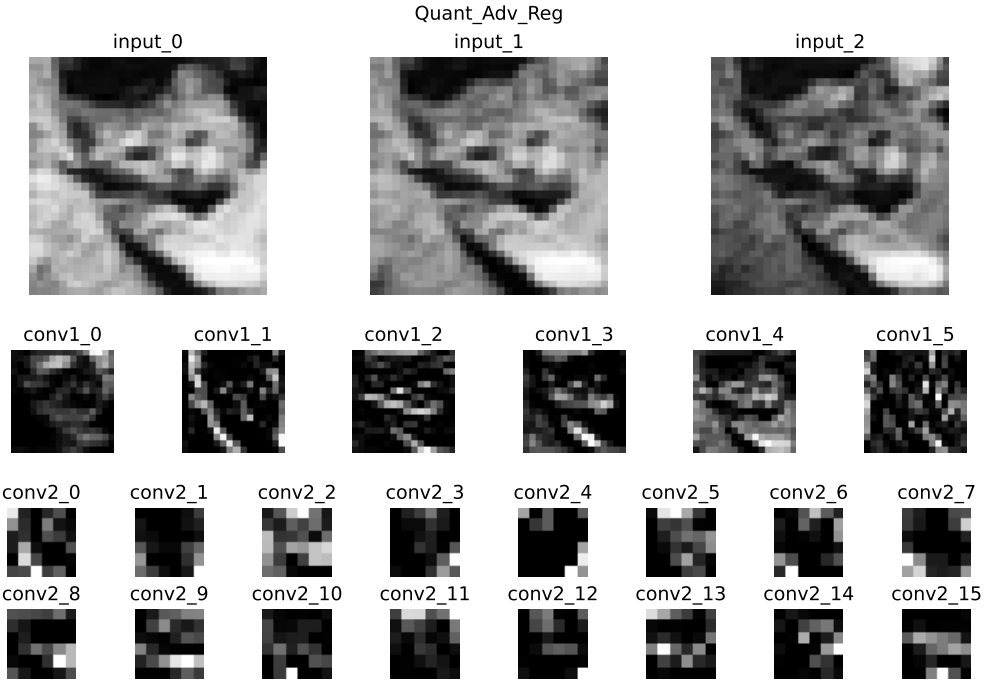

图 20: 量化 + 对抗训练 + 正则化模型特征图

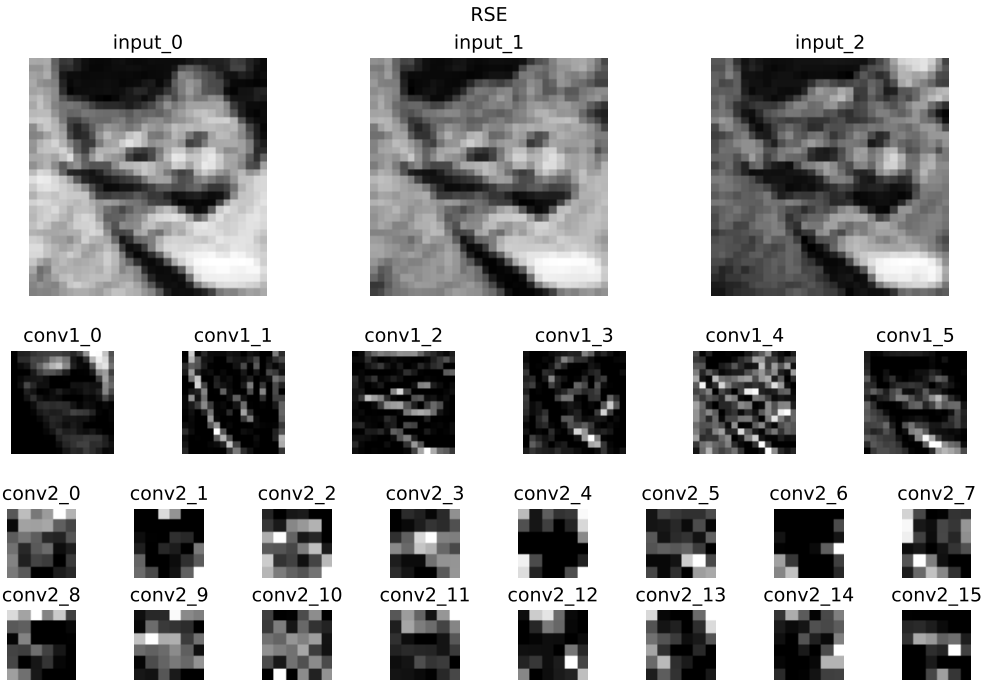

图 21: 随机自集成模型特征图