影音总结文档-深度学习

- 金国军

背景

在高清化数字化之前，很多的影像资料是以胶片、磁带的方式拍摄、存放。随着保存时间的增加，这些影像资料出现了不同程度损坏，质量下降，甚至严重到不能使用情况。而这些旧的影像资料又具有极高的价值，对其做修复、提清，具有很大的市场空间。

因为这些影像资料场景复杂多变、质量又比较差，较传统的方法修复需要非常大的人力投入。深度学习技术发展日新月异，特别是在计算机视觉学上的突破性进展，为旧影像资料的修复提供了可能。我们尝试以深度学习做图像、视频的超清化(super resolution)为切入点，做旧的影像资料的修复，以期达到较理想的视觉效果。

本文档是对之前验证工作的总结，包括在深度学习在影像超清化的验证总结，影像资料损坏、修复效果的理论分析与部分验证效果。

深度学习验证总结

1 深度学习简介

深度学习(DeepLearning, DL)可以理解为人工神经网络的发展，因网络层增多、加深，该方法命名为深度学习，它是机器学习的一个分支。受算力、数据量、训练困难等诸多方面的影响，人工神经网络在开始一段时间慢慢淡化。在 2012 年 ImageNet 图像识别比赛上，深度学习将效果大幅度提升，随后在学界、工业界掀起热潮。

1.1 深度学习原理

深度学习采用神经网络结构，包括输入部分、神经网络、输出，训练时计算输出与目标之间的差值(loss)，通过反向传播(BackPropagation，BP)算法修正网络中的参数减少 loss，优化网络，从而达到神经网络的输出与目标尽可能相近的目标。

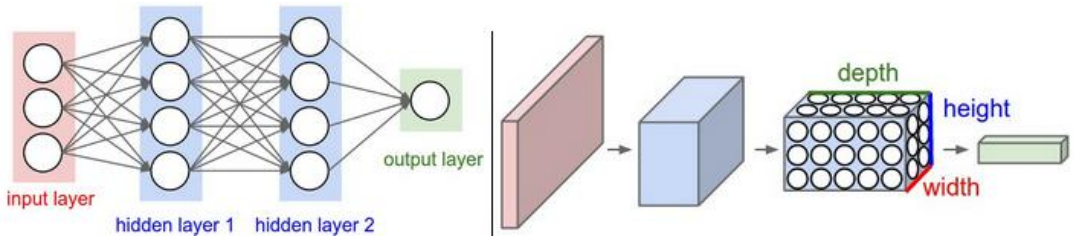


图 1.1.1 深度学习基本网络结构图

深度学习的背后是 BP 算法，以来自[5]的一个简单实例做说明：假设有如下网络中有一结构如下，设定了各个节点的初始化权重值，当输入数值是 0.05 和 0.1 时，期待的输出为 0.01 和 0.99。

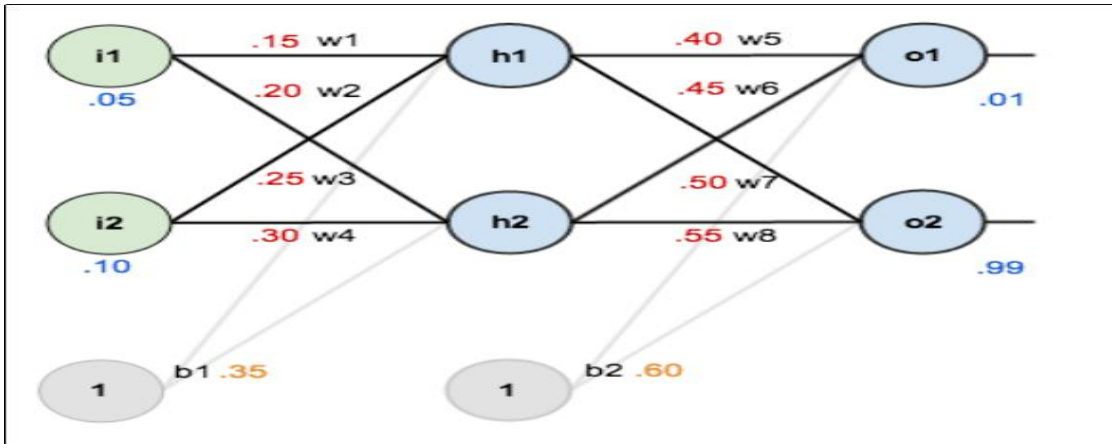


图 1.1.2 简单网络结构，并设定初始化

根据前向传输(ForwardPass)，在有输入的情况下，可以得到对应的输出是 0.751 和 0.773，可以看到输出和期望之间存在一定的差距，根据（loss 公式）我们可以得到之间的差（loss）。接下来就是使用 BP 算法来更新网络中的权重参数，以 w5 参数为例，图形表示为：

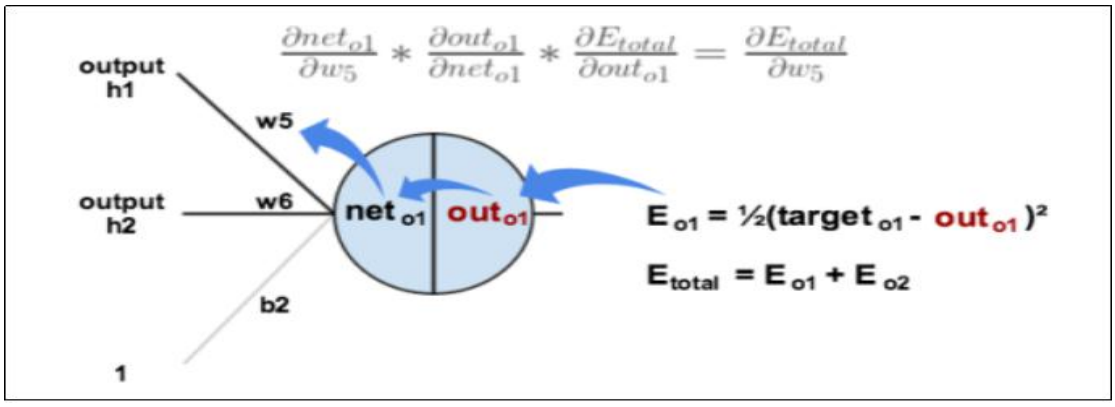


图 1.1.3 单个节点更新过程

整个网络中的权重更新，可以采用同样的方式，依此类推。

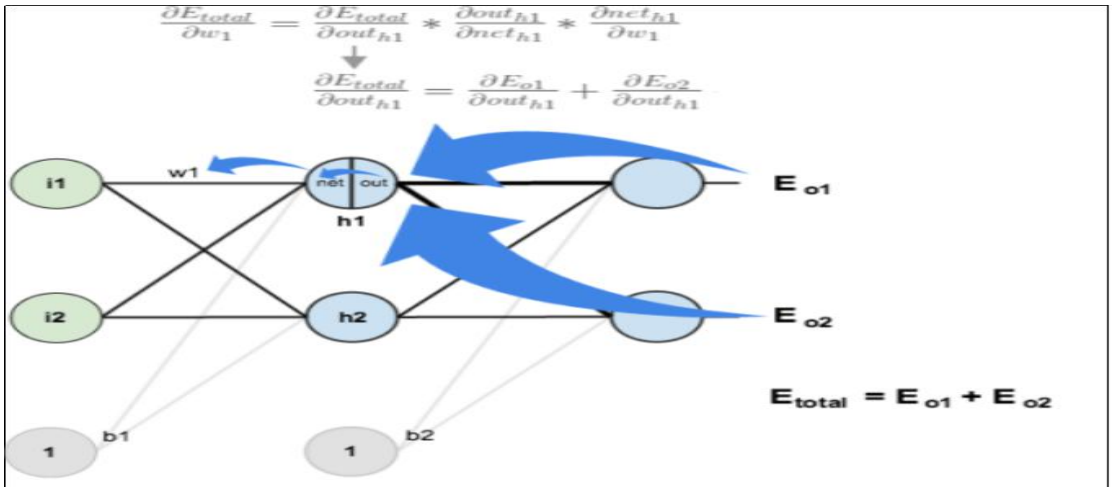


图 1.1.4 整个网络中 BP 算法更新参数过程

一般而言深度学习的最终效果与训练集的大小、网络的大小成正比。也就是说，训练集越多，覆盖到的场景就越多，所包含的特征越多，训练的效果就会越多；深度学习的网络越大，内部的参数越多，就能“存储”更多的信息量，效果也就会越好。

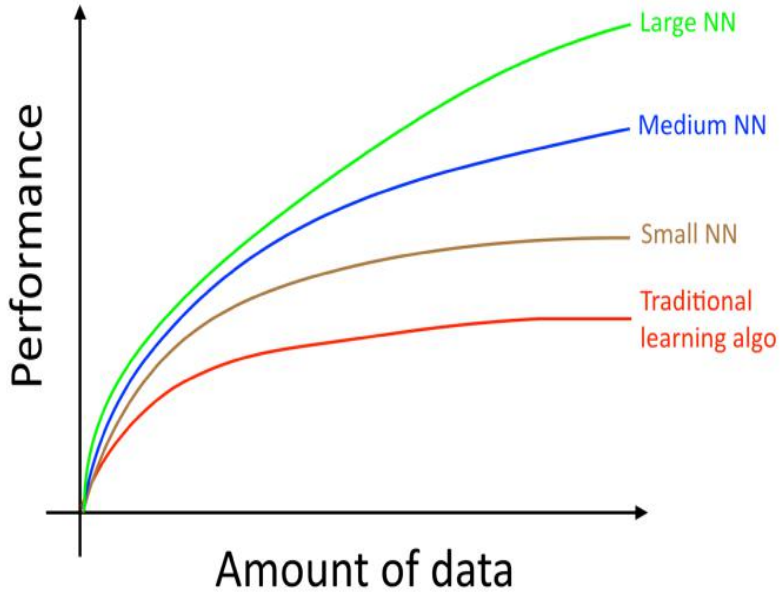


图 1.1.5，深度学习效果与训练集大小、网络大小的关系图

从图可以看出，获取更多的训练集需要更大的投入，而且训练集增多，效果的提升会变缓。网络的越大，对机器资源的要求越高，训练时间越长，训练难度就会越大，而相应的效果提升也会变缓。所以，最终的训练集和网络大小的选取，是效果与投入、难易之间的折中考虑。

1.2 深度学习术语

卷积网络(ConvolutionalNet,ConvNet)、网络层(layer)：常见的深度学习中的网络结构为卷积网络，由一层层的网络层组成，常见的层包括：卷积网络层(ConvolutionalLayer,Conv)、批量正则化层(BatchNormalization, BN)、池化层(Pool)、ReLU(属于激活层)等。

. 卷积过程

网络执行过程，每层卷积网路的输入、输出具有如下的关系

- 输入：size W1×H1×D1
- 卷积层：卷积和 K
  卷积的数量 F
  卷积步长 S
  填充 P.
- 输出：W2=(W1−F+2P)/S+1
  H2=(H1−F+2P)/S+1
  D2=K

斯坦福大学 cs231 课程[2]中，提供了卷积层执行过程的动态图，非常直观。

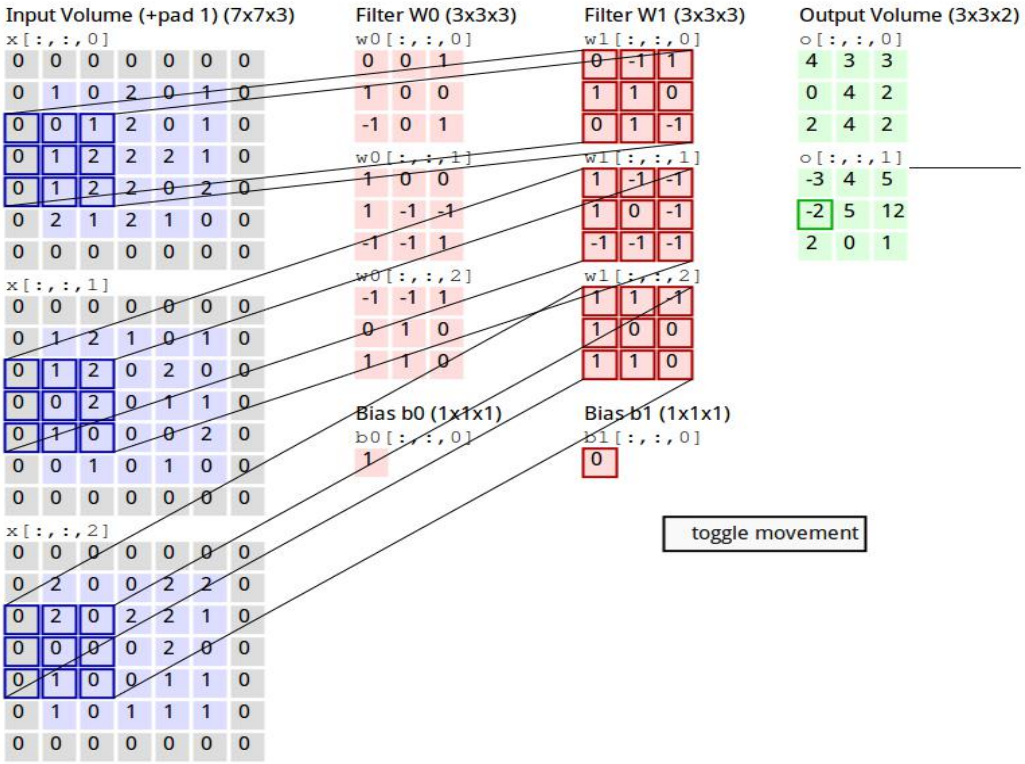


图 1.2.1 卷积过程基本图

. DataAugment

深度学习主要对训练数据集的要求非常高，而很多情况下，训练集的数量有限，通过 dataAugment 的方法，较数据集扩展，充分利用数据集中的特征。如，图像做翻转、旋转，截取任意区域。

. 过拟合与欠拟合

如果训练集数量有限，或者场景简单而网络设计的网络，训练之后，网络可能在训练集上有非常好的表现，而到了测试场景中，效果却比较差，这种情况称为过拟合；如果训练集比较大，场景比较多，而设计的网络比较简单，这种情况下，网络不能有效的覆盖训练集的特征，即使在训练集上都没有好的结果，这成为欠拟合。

. 预训练与初始化

网络训练受很多因素影响，而初始时的状态影响训练的速度和最终的效果。深度学习最开始的几年，多采用预训练的方式，也就是采用较小的数据集，较大一些的学习率先预训练网络；现在常采用一些较好的初始化算法做初始化，包括高斯初始化，Xavier 初始化、Kaiming（MSRA）初始化

. 网络结构

神经网络多是由层来构建，研究人员构造了不同的网络结构，带来了一定的效果提升；而网络结构比较有代表的，主要有 VGG、GoogleNet、EncoderDecoder、残差网络，长连接等。

BatchNormalization：批正则化，通过将 activation 规范为均值和方差的手段使得原本会减小的 activation 的 scale 变大，从而防止“梯度消散”。在生成网络中，我们会使用，但是也看到，想 EDSR 论文也验证，有没有 BN，超清化效果相差不大，而且没有 BN，能节省

内存空间[10]。

BN 在 classification 当中的效果较好，主要原因是因为 classification 对于 scale 不敏感。但是 superresolution 这样图片到图片的变换，per image 的 scale 还是一个有效信息。类似的一个情形是 style transfer，同样也是图片到图片的变换，Ulyanov et al.提出用 instance normalization 的方法，IN 可以理解成为每个图片自己做 BN，这样比 BN 能保留更多 scale 信息。更新的研究表明如果训练收敛不是问题的话，进一步去掉 IN 的效果也会更好。总的来说 BN 对于简化调参有好处（更容易收敛），但是并不一定真的适合所有应用(https://www.zhihu.com/question/62599196)。

. 迁移学习

基本思想是，在已训练好的网络（训练库）上，通过很小的调整、很小的训练，能够在新的任务上有比较好的效果。比如，已有识别 ImageNet 图像集的 VGG 网络，网络的前面部分主要功能是提取图像的特征，而后面很小部分主要是对这些特征做分类；通过迁移后面的分类部分，以很小的代价，就可以达到比较好的效果。

斯坦福 cs231 课程[2]以及一些博客[1]，提供深度学习更多基础性的介绍。

2. 平台工具

2.1 系统平台

英伟达(NVIDIA)的 GPU 在内存、浮点计算上都具有非常强的能力，同时英伟达公司为深度学习提供更好、更多的技术平台和解决方案；当前来看，英伟达 GPU 是深度学习的硬件首选。

当前比较流行的深度学习框架包括 caffe、torch、tensorflow、keras 等，虽然在 Windows 平台上可以搭建这些框架，但是比较麻烦，而且会存在各种问题，从方便性、普及性等角度考虑，我们以 Ubuntu16.04 操作系统作为我们的系统平台。Ubuntu 平台上 Tesla-GPU 驱动的安装，请参考附件[A.1]。

2.2 框架

我们验证过程主要使用了 caffe、Torch(pyTorch)、Tensorflow 这三个框架[3]。

2.2.1 Caffe

第一个主流的工业级深度学习工具，它开始于 2013 年底，由 UC Berkely 的 Yangqing Jia 编写和维护的具有出色的卷积神经网络实现。在计算机视觉领域 Caffe 依然是最流行的工具包。它有很多扩展，但是由于一些遗留的架构问题，不够灵活且对递归网络和语言建模的支持很差。

【http://ethereon.github.io/netscope/#/editor】提供工具，可视化 caffe 的网络结构，安装参考附件[A.2]。

2.2.2 Torch

Facebook 力推的深度学习框架，主要开发语言是 C 和 Lua。有较好的灵活性和速度。它实现并且优化了基本的计算单元，使用者可以很简单地在此基础上实现自己的算法，不用浪费精力在计算优化上面。核心的计算单元使用 C 或者 cuda 做了很好的优化。在此基础之上，使用 lua 构建了常见的模型。Torch 是我们试验测试中使用最多的框架。

print(net)可输出网络结构。安装参考附件[A.3]。

2.2.3 TensorFlow

Google 开源的其第二代深度学习技术——被使用在 Google 搜索、图像识别以及邮箱的深度学习框架。是一个理想的 RNN（递归神经网络）API 和实现，TensorFlow 使用了向量运算的符号图方法，使得新网络的指定变得相当容易，支持快速开发。缺点是速度慢，内存占用较大。（比如相对于 Torch）

自带工具 Tensorboard 可视化网络结构，内部参数等，但是因使用少，一直不习惯，安装参考附件[A.4]。

更多的深度学习框架比较，请参考：[6-9]

3. 理论支持

我们的实验验证，前期主要基于已有的公开的研究工作、项目基础上，这样会有一个比较理想的预期目标，同时论文中提供的参数、训练过程等，可以提高我们训练的速度。本节介绍后续验证中参考的主要论文、项目。

3.1 基础论文

3.1.1 VGG

VGG[23]中的预处理：图片的预处理就是每一个像素减去了均值，算是比较简单的处理；卷积核：整体使用的卷积核都比较小（3x3），3x3 是可以表示「左右」、「上下」、「中心」这些模式的最小单元了。还有比较特殊的 1x1 的卷积核（Inception-v1 也有这个东西），可看做是空间的线性映射。

VGG 中前面几层是卷积层的堆叠，后面几层是全连接层，最后是 softmax 层。所有隐层的激活单元都是 ReLU，论文中会介绍好几个网络结构，只有其中一个应用了局部响应归一化层（Local Response Normalisation）。

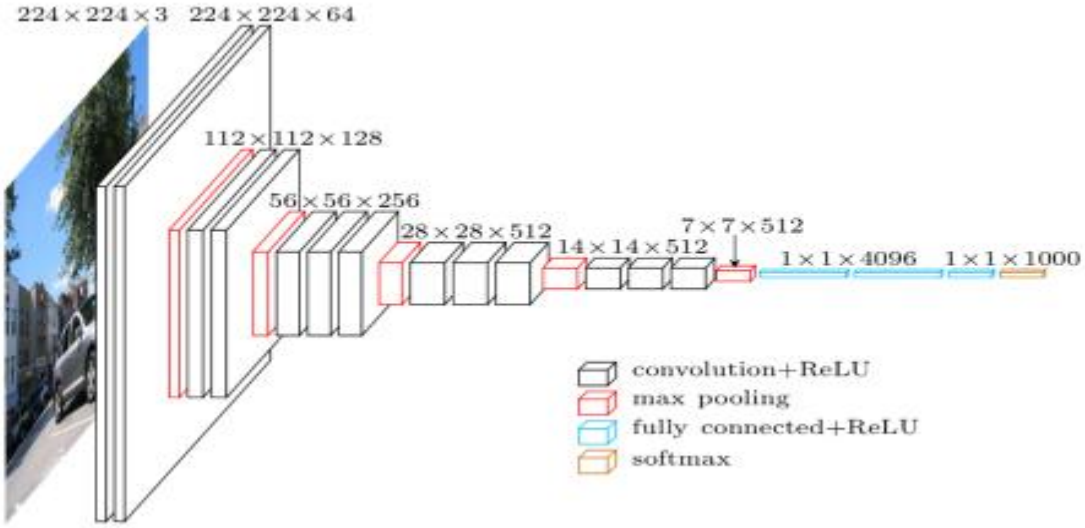


图 3.1.1 VGG19 的结构示意图

使用多个较小卷积核的卷积层代替一个卷积核较大的卷积层，一方面可以减少参数，另一方面作者认为相当于是进行了更多的非线性映射，可以增加网络的拟合/表达能力。

3.1.2 ResNet

残差网络[24]中的残差块可选用的结构图：

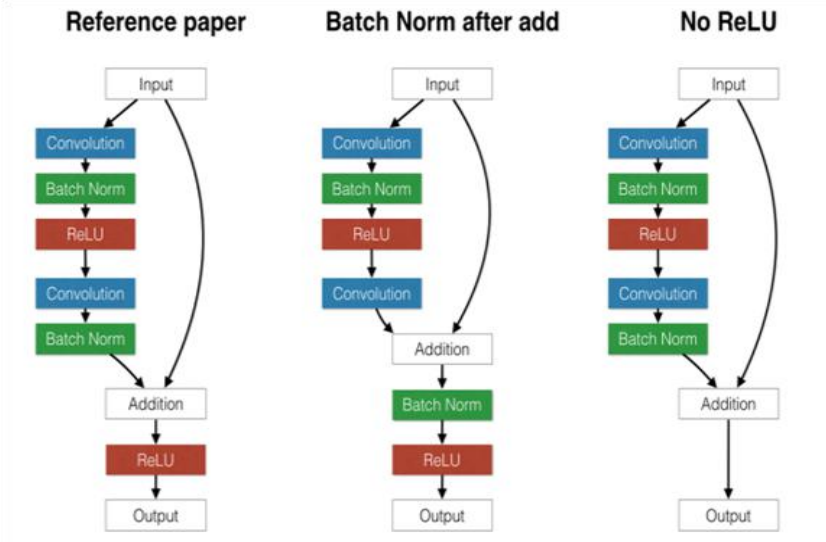


图 3.1.2 ResNet 中使用的网络结构图

根据输入将层表示为学习残差函数。实验表明，残差网络更容易优化，并且能够通过增加相当的深度来提高准确率。核心是解决了增加深度带来的副作用（退化问题），这样能够通过单纯地增加网络深度，来提高网络性能。

作者在 ImageNet 上实验了一个 152 层的残差网络，比 VGG 深 8 倍，取得了 3.57%的错误率；通过一系列实验证明了表示的深度（即网络的深度）对很多视觉识别任务都至关重要。仅仅由于使用了非常深的网络，作者就在 COCO 目标检测数据集上获得了 28%的相对提升。

3.1.3 GAN

生成对抗网络[25]主要思想是拥有两个竞争的神经网络模型。 一个将噪声数据作为输入，并产生样本（所谓的生成器）。 另一个模型（称为判别器）从生成器和训练数据接收样本，并且必须能够区分两个来源。 这两个网络进行连续的博弈，生成器学习产生越来越多的现实样本，鉴别器正在学习越来越好地区分生成的数据和实际数据。 这两个网络同时进行训练，最后的希望是竞争能够使生成器生成的样本与实际数据不可区分。

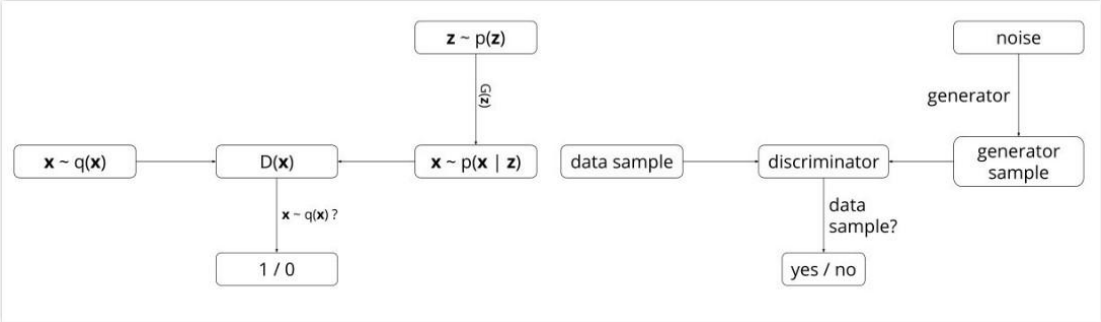


图 3.1.3 生成对抗网络算法框图

GAN 优点：

- 根据实际的结果，它们看上去可以比其它模型产生了更好的样本（图像更锐利、清晰）。
- 生成对抗式网络框架能训练任何一种生成器网络（理论上-实践中，用 REINFORCE 来训练带有离散输出的生成网络非常困难）。大部分其他的框架需要该生成器网络有一些特定的函数形式，比如输出层是高斯的。重要的是所有其他的框架需要生成器网络遍布非零质量（non-zero mass）。生成对抗式网络能学习可以仅在与数据接近的细流形（thin manifold）上生成点。
- 不需要设计遵循任何种类的因式分解的模型，任何生成器网络和任何鉴别器都会有用。
- 无需利用马尔科夫链反复采样，无需在学习过程中进行推断（Inference），回避了近似计算棘手的概率的难题。

GAN 缺点：

- 解决不收敛（non-convergence）的问题。

目前面临的基本问题是：所有的理论都认为 GAN 应该在纳什均衡（Nash equilibrium）上有卓越的表现，但梯度下降只有在凸函数的情况下才能保证实现纳什均衡。当博弈双方都由神经网络表示时，在没有实际达到均衡的情况下，让它们永远保持对自己策略的调整是可能的【OpenAI Ian Goodfellow 的 Quora】。

- 难以训练：崩溃问题（collapse problem）

GAN 模型被定义为极小极大问题，没有损失函数，在训练过程中很难区分是否正在取得进展。GAN 的学习过程可能发生崩溃问题（collapse problem），生成器开始退化，总是生成同样的样本点，无法继续学习。当生成模型崩溃时，判别模型也会对相似的样本点指向相似的方向，训练无法继续。【Improved Techniques for Training GANs】

- 无需预先建模，模型过于自由不可控。

与其他生成式模型相比，GAN 这种竞争的方式不再要求一个假设的数据分布，即不需要 formulate p(x)，而是使用一种分布直接进行采样 sampling，从而真正达到理论上可以完全逼近真实数据，这也是 GAN 最大的优势。然而，这种不需要预先建模的方法缺点是太过自由了，对于较大的图片，较多的 pixel 的情形，基于简单 GAN 的方式就不太可控了。在 GAN[25] 中，每次学习参数的更新过程，被设为 D 更新 k 回，G 才更新 1 回，也是出于类似的考虑。

3.2 超清化论文

3.2.1 SRCNN

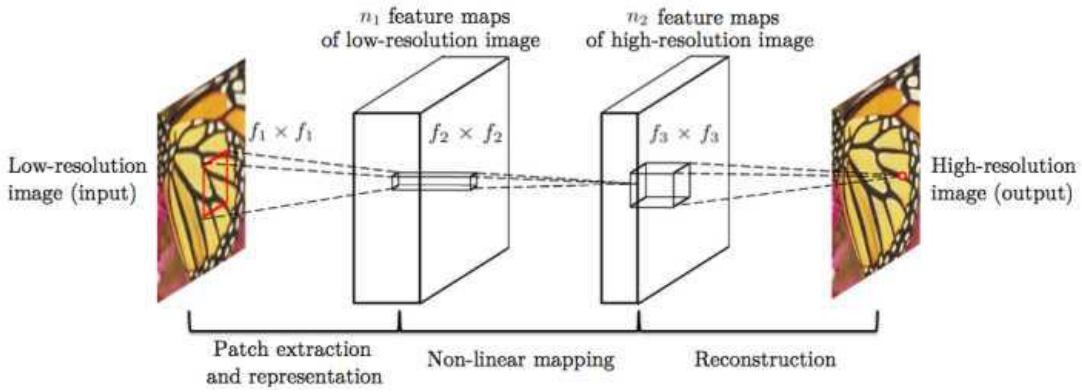


图 3.2.1 SRCNN 算法结构图

首个应用于图像超清问题的 CNN 网络结构，算法发输入为低清图像，输出为高清图像。该结构分为三个步骤：低清图像的特征抽取、低清特征到高清特征的映射、高清图像的重建。图像超清问题的特点在于，低清图像和高清图像中很大部分的信息是共享的，基于这个前提，在 CNN 出现之前，业界的解决方案是使用一些特定的方法，如 PCA、Sparse Coding 等将低分辨率和高分辨率图像变为特征表示，然后将特征表示做映射。

基于传统的方法结构，CNN 也将模型划分为三个部分，即特征抽取、非线性映射和特征重建。由于 CNN 的特性，三个部分的操作均可使用卷积完成。因而，虽然针对模型结构的解释与传统方法类似，但 CNN 却是可以同时联合训练的统一体，在数学上拥有更加简单的表达。

不仅在模型解释上可以看到传统方法的影子，在具体的操作上也可以看到。在上述模型中，需要对数据进行预处理，抽取出很多 patch，这些 patch 可能互有重叠，将这些 Patch 取合集便是整张图像。上述的 CNN 结构是被应用在这些 Patch 而不是整张图像上，得到所有图像的 patch 后，将这些 patch 组合起来得到最后的高清图像，重叠部分取均值。

开源项目 Waifu2x[22]以该论文为基础，后续增加了网络的大小，提供了图像的超清化、去噪等功能。实际验证发现，对于动画图像具有比较好的提清效果，而对于真实图像的超清化，则没有比较明显的效果。

3.2.2 VDSR

SRCNN 的方法虽然效果远高于传统方法，但是却有若干问题：

- 训练层数少，没有足够的视野域；
- 训练太慢，导致没有在深层网络上得到好的效果；
- 不能支持多种倍数的高清化。

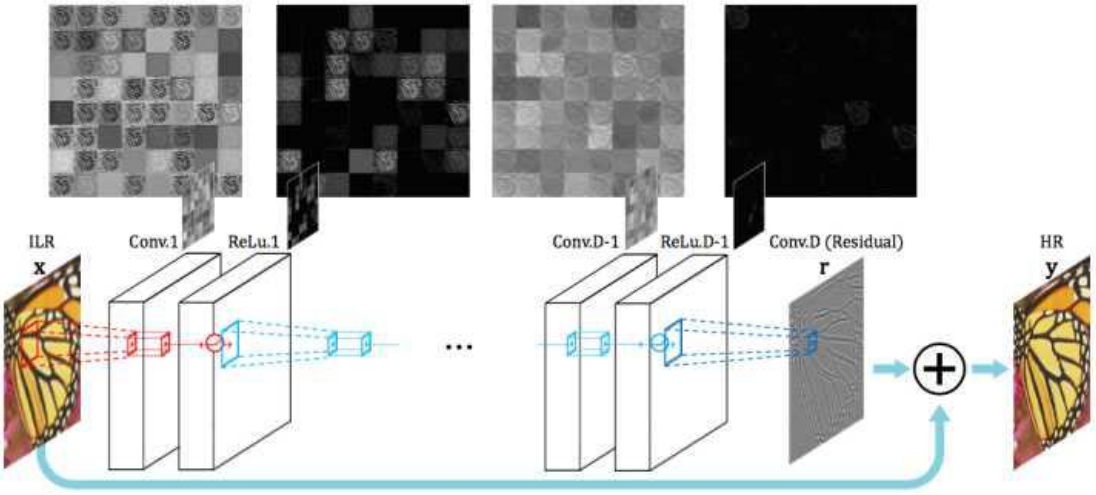


图 3.2.2 VDSR 算法结构

针对上述问题，VDSR[23]算法，提出了采用更深的网络模型,并用三种技术解决了 SRCNN 的问题。

1. 残差学习:

   CNN 是端到端的学习，如果像 SRCNN 那样直接学习，那么 CNN 需要保存图像的所有信息，需要在恢复高清细节的同时记住所有的低分辨率图像的信息。如此，网络中的每一层都需要存储所有的图像信息，这就导致了信息过载，使得网络对梯度十分敏感，容易造成梯度消失或梯度爆炸等现象。而图像超清问题中，CNN 的输入图像和输出图像中的信息很大一部分是共享的。残差学习是只针对图像高清细节信息进行学习的算法。如上图所示，CNN 的输出加上原始的低分辨率图像得到高分辨率图像，即 CNN 学习到的是高分辨率图像和低分辨率图像的差。如此，CNN 承载的信息量小，更容易收敛的同时还可以达到比非残差网络更好的效果。高清图像之所以能够和低清图像做加减法，是因为，在数据预处理时，将低清图像使用插值法缩放到与高清图像同等大小。于是虽然图像被称之为低清，但其实图像大小与高清图像是一致的。

2. 高学习率

   在 CNN 中设置高学习率通常会导致梯度爆炸，因而在使用高学习率的同时还使用了自适应梯度截断。截断区间为[- θ / γ， θ / γ]，其中 γ 为当前学习率， θ 是常数。

3. 数据混合

   最理想化的算法是为每一种倍数分别训练一个模型，但这样极为消耗资源。因而，同之前的算法不同，本技术将不同倍数的数据集混合在一起训练得到一个模型，从而支持多种倍数的高清化。

3.2.3 Perceptual-SR

在此之前，使用 CNN 来解决高清问题时，对图像高清化的评价方式是将 CNN 生成模型产生的图像和实际图像以像素为单位计算损失函数（一般为欧式距离）。此损失函数得到的模型捕捉到的只是像素级别的规律，其泛化能力相对较弱。

而感知损失[14]，则是指将 CNN 生成模型和实际图像都输入到某个训练好的网络中，得到这两张图像在该训练好的网络上某几层的激活值，在激活值上计算损失函数。由于 CNN 能够提取高级特征，那么基于感知损失的模型能够学习到更鲁棒更令人信服的结果。

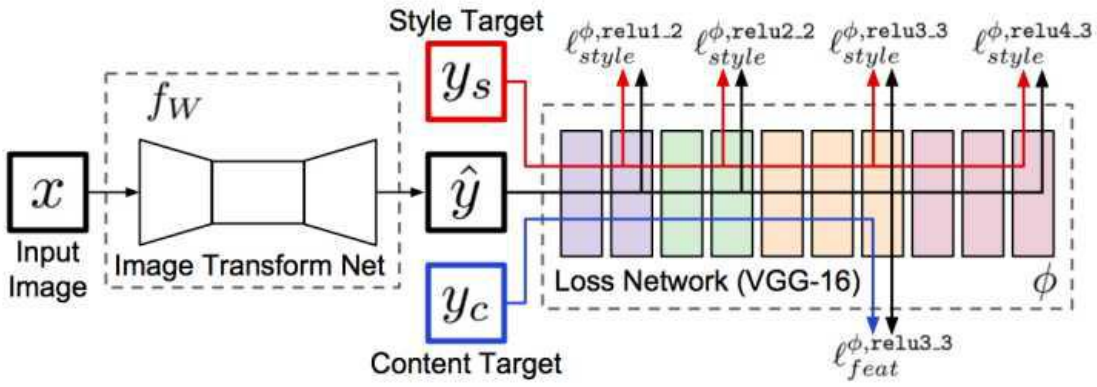


图 3.2.3. 基于感知损失的图像风格转换网络

该网络也可用于图像超清问题。左侧是一个待训练的转换网络，用于对图像进行操作；右侧是一个已训练好的网络，将使用其中的几层计算损失。

Perceptual-SR 为感知损失网络，该网络本是用于快速图像风格转换。在这个结构中，需要训练左侧的 Transform 网络来生成图像，将生成的图像 Y 和内容图像与风格图像共同输入进右侧已经训练好的 VGG 网络中得到损失值。如果去掉风格图像，将内容图像变为高清图像，将输入改为低清图像，那么这个网络就可以用于解决图像超清问题了。

3.2.4 SRGAN

对抗神经网络称得上是近期机器学习领域最大的变革成果。其主要思想是训练两个模型 G 和 D。G 是生成网络而 D 是分类网络，G 和 D 都用 D 的分类准确率来进行训练。G 用于某种生成任务，比如图像超清化或图像修复等。G 生成图像后，将生成图像和真实图像放到 D 中进行分类。使用对抗神经网络训练模型是一个追求平衡的过程：保持 G 不变，训练 D 使分类准确率提升；保持 D 不变，训练 G 使分类准确率下降，直到平衡。GAN 框架使得无监督的生成任务能够利用到监督学习的优势来进行提升。基于 GAN 框架，只要定义好生成网络和分类网络，就可以完成某种生成任务。

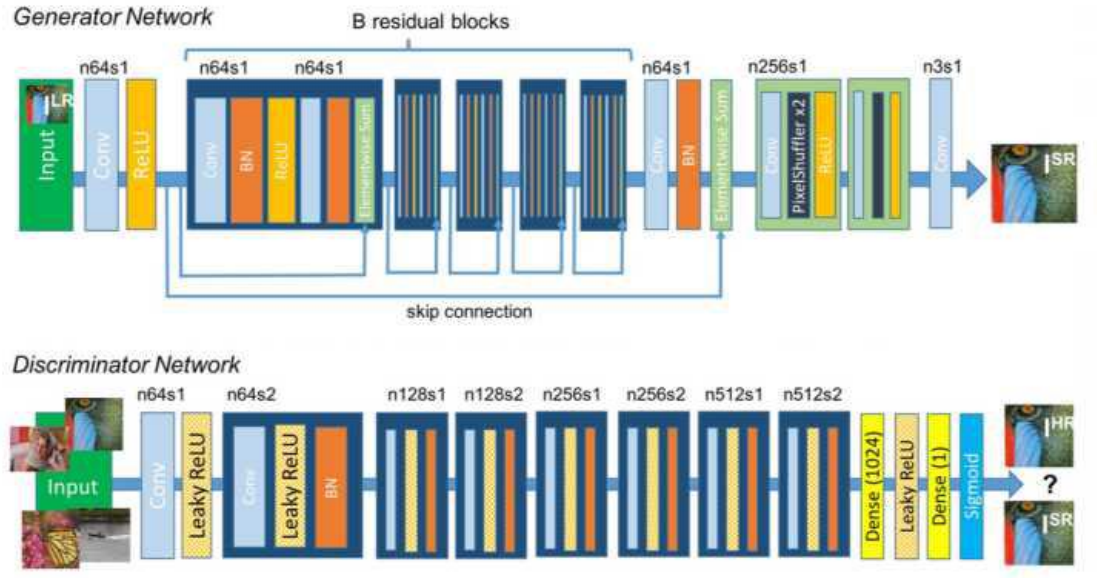


图 3.2.4 SRSAN 算法结构图

对抗训练的生成网络 G 和判别网络结构 D

而将 GAN 应用到图像高清问题的这篇论文[15]，可以说是集大成之作，算法中的网络结构如上图所示，上半部分是生成网络 G，层次很深且使用了 residual block 和 skip-connection 结构；下半部分是判别网络 D。生成模型层次深且使用了 residual block 和 skip-connection；在 GAN 的损失函数的基础上同时添加了感知损失。

其中，生成网络自己也可以是一个单独的图像超清算法。论文中分析了 GAN 和 non-GAN 的不同，发现 GAN 主要在细节方面起作用，但无法更加深入地解释。“无法解释性”也是 GAN 目前的缺点之一。

3.2.5 EnhanceNet

Output size	Layer
$w \times h \times c$	Input I_{LR}
$w \times h \times 64$	3x3 Conv, ReLU
	Residual: Conv 3x3, ReLU, Conv 3x3
	...
	Residual: Conv 3x3, ReLU, Conv 3x3
$2w \times 2h \times 64$	2x nearest neighbor upsampling 3x3 Conv, ReLU
$4w \times 4h \times 64$	2x nearest neighbor upsampling 3x3 Conv, ReLU
	3x3 Conv, ReLU
$4w \times 4h \times c$	3x3 Conv
	Residual image I_{res}
	Output $I_{\text{est}} = I_{\text{bicubic}} + I_{\text{res}}$

图 3.2.5 EnhanceNet 中生成部分结构

EnhanceNet[17]网络的结构、Loss 等都近似 SRGAN，两者之间的不同之处主要包括：

1. EnhanceNet 中的长连接是从网络的 input 之后，连接到网络的 output 之前，这样可以在非常大的程度上保证输出图像的颜色保持不变；
2. 评估 loss 除了 SRGAN 中的 perceptual-loss(P)和 Adversarial-loss(A)外，EnhanceNet 中还使用了 texture loss(T)，并且权重的大小与 SRGAN 中的权重的大小也有所不同

3.2.6 EDSR

NTIRE 2017 超分辨率挑战赛冠军方法 EDSR[16]，该方法使用的技术可以总结为

- 更大的网络，网络层使用的通道数由常用的 64 增加到 256，网络中的残差块由常见的 10 或 16 增加到 32
- 残差块中去掉 BatchNormalization，既减少了内存开销、计算量，效果又很好
- 残差块中使用了缩放，参数为 0.1，用来保证训练过程的稳定

- loss 评估函数由常用的 L2 改为 L1
- 训练中的训练集由比赛主办方提供，采用 matlab 的 bicubic 下采样和预训练的神经网络生成。Matlab 的 bicubic 做图像的下采样，做了图像的反锯齿纹操作

我们分析认为，EDSR 之所以在超清化比赛中能够获得冠军，很大程度上是因为使用的网络大，这个大的网络，训练使用了 14 天

3.2.7 VideoSR

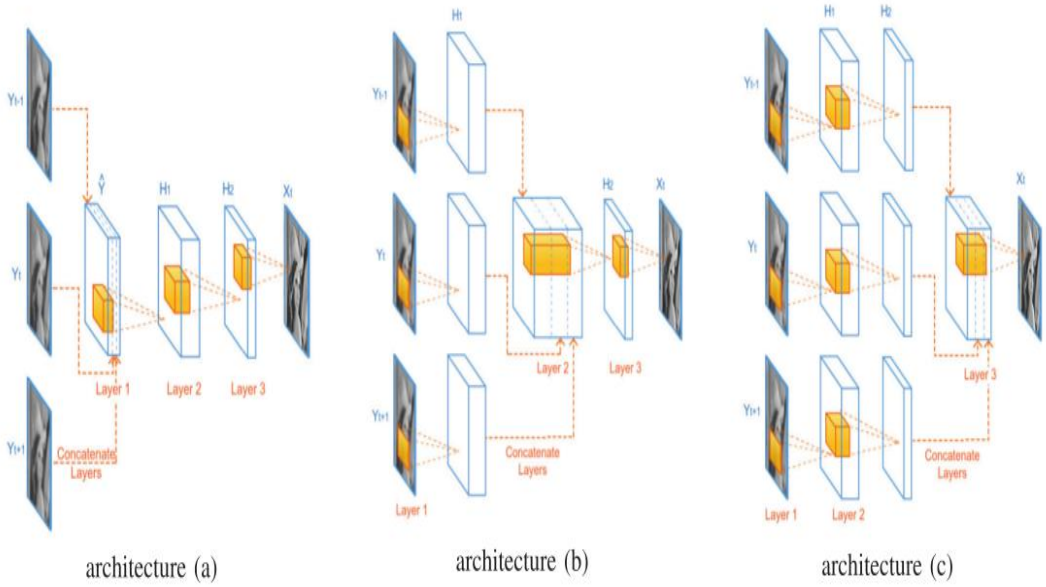


图 3.2.7 多帧输入做图像超清化的网络结构

论文[19]思路是对视频的超清化，我们可以输入连续的多帧，利用这多帧内的信息；除此之外，因为多帧的图像不是完全一致的，输入的图像需要运动补偿。VideoSR 采用的方式是用现有的算法做运动估计；而论文[20-21]则是将运动估计也放在了神经网络的训练过程中，从而学习到如何更好的运动估计，以得到更好的 SR 效果。

3.3 图像目标识别论文

图像目标识别任务，既要把图中的物体识别出来，又要用方框框出它的位置。目标识别会存在几种方法，但是就实际效果与思路来看，Kaiming He 和 Ross Girshick 等人提出的以 RCNN 结合 region proposal 和 CNN 分类的方法，其技术发展路线为：RCNN -> SppNET ->Fast-RCNN -> Faster-RCNN。

3.3.1 RCNN

RCNN(Region CNN)[28]，其基本思想是取不同的大小的“框”，让框出现在不同的位置，得出这个框的判定得分，取得分最高的那个框。为了能将识别位置尽可能的准确，需要将所有的位置都尽可能的“框”一遍，依次从左上角扫到右下角取位置，但是这样太暴力，计算量太大。于是考虑采用一些选择性

算法，将一些重复的”框”，比如在别的框内部的小框等去掉。

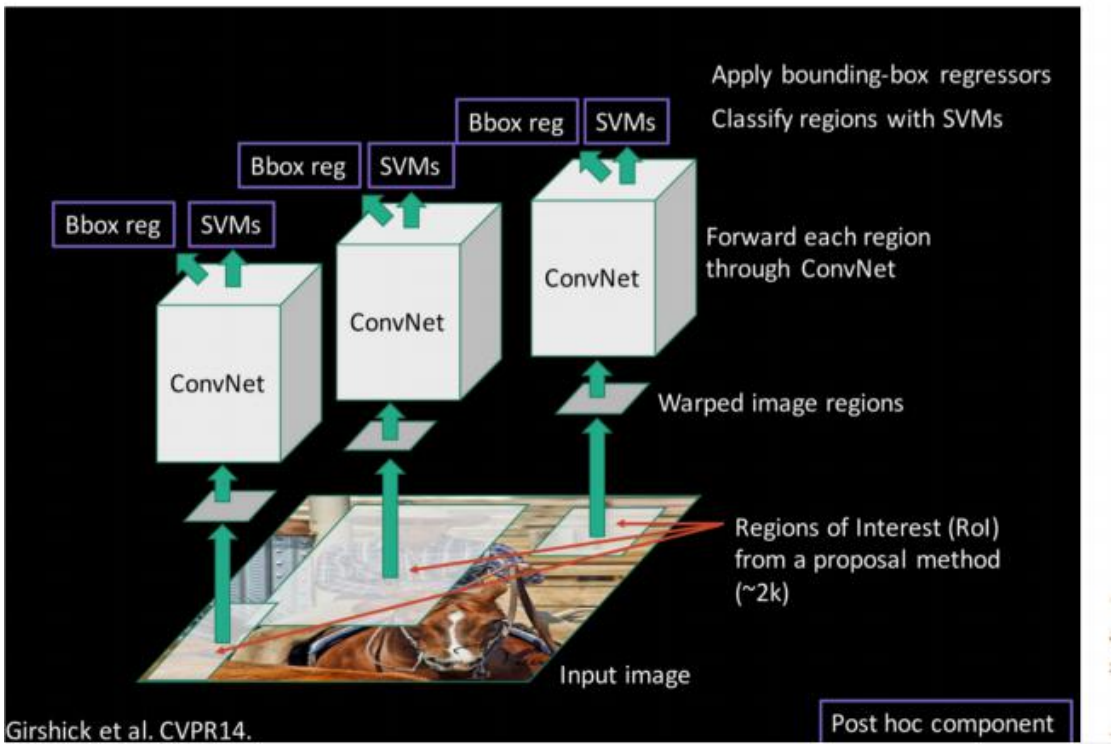


图 3.3.1 RCNN 基本过程

RCNN 基本过程为：

1. 训练（或者下载）一个分类模型（比如 AlexNet）
2. 对该模型做 fine-tuning，比如改为 20 个分类，去掉最后一个全连接
3. 特征提取，提取图像的所有候选框（选择性搜索），对于每一个区域计算得到特征
4. 训练一个 SVM 分类器（二分类）来判断这个候选框里物体的类别
5. 使用回归器精细修正候选框位置

3.3.2 SppNet

Spatial Pyramid Pooling[29]（空间金字塔池化），有两个特点：

1. 结合空间金字塔方法实现 CNNs 的对尺度输入：将金字塔思想加入到 CNN，实现了数据的多尺度输入。

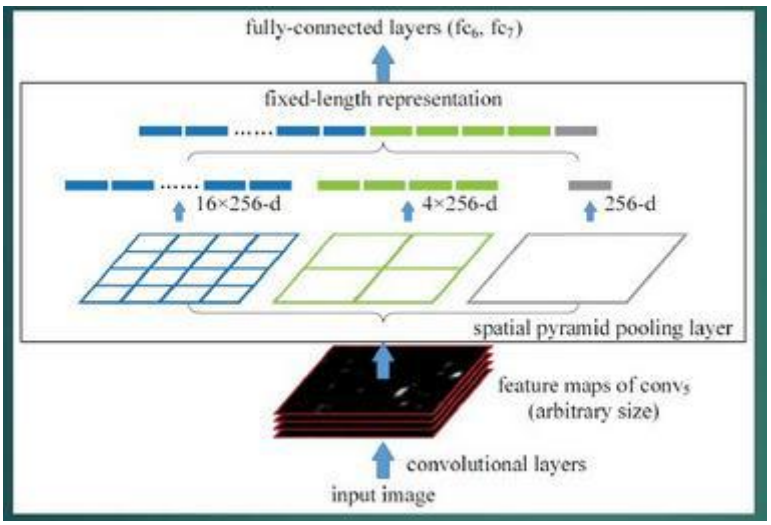


图 3.3.2 SppNet，在卷积层和全连接层之间加入了 SPP layer。此时网络的输入可以是任意尺度的，在 SPP layer 中每一个 pooling 的 filter 会根据输入调整大小，而 SPP 的输出尺度始终是固定的

2. 只对原图提取一次卷积特征：只对原图进行一次卷积得到整张图的 feature map，然后找到每个候选框 zaifeature map 上的映射 patch，将此 patch 作为每个候选框的卷积特征输入到 SPP layer 和之后的层。节省了大量的计算时间，比 R-CNN 有一百倍左右的提速。

3.3.3 Fast R-CNN

RCNN 使用选择性搜索等预处理步骤来提取潜在的“框”作为输入，但是 RCNN 仍会有严重的速度瓶颈，因为对所有 region 进行特征提取时会有重复计算。Fast R-CNN[30]提出了一个可以看做单层 sppnet 的网络层，叫做 ROI Pooling，这个网络层可以把不同大小的输入映射到一个固定尺度的特征向量。

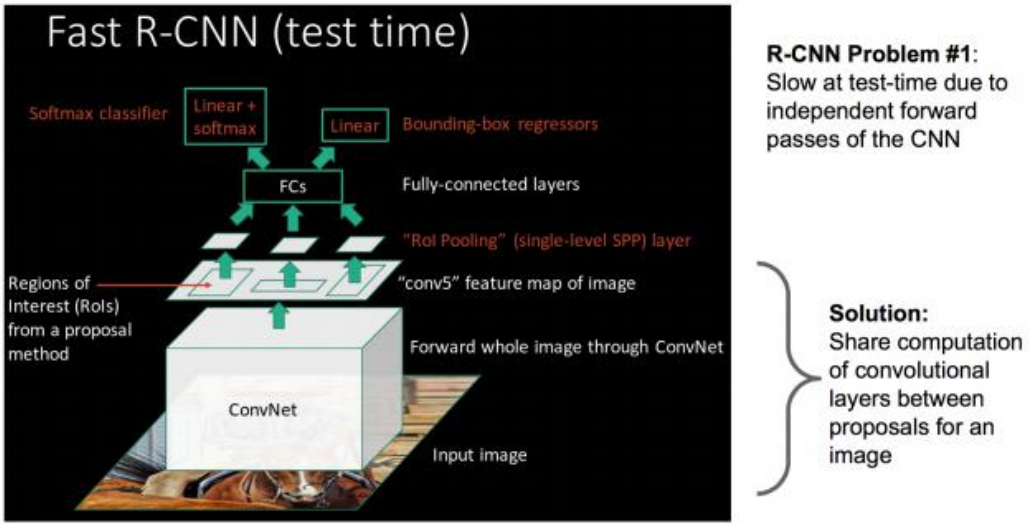


图 3.3.3 Fast R-CNN 算法基本流程。算法巧妙的把 bbox regression 放进了神经网络内部，与区域分类和并成为了一个 multi-task 模型，而不需要想 RCNN 那样，先提取候选区域，然后 CNN 提取特征，最后 SVM 分类。

3.3.4 Faster R-CNN

Fast R-CNN 存在的问题：存在瓶颈：选择性搜索，找出所有的候选框，这个也非常耗时，Faster R-CNN[31]加入一个提取边缘的神经网络，也就说找到候选框的工作也交给神经网络来做了。做这样的边缘提取任务的神经网络叫做 Region Proposal Network(RPN)。这使得检测速度大幅度提升。

其基本思想是：在提取好的特征图上，对所有可能的候选框进行判别。由于后续还有位置精修步骤，所以候选框实际比较稀疏。该算法过程如下图 3.3.4，图中的 raw feature extration net 模块做输入图像的原始特征提取，包含若干层 conv+relu，直接套用 ImageNet 上常见的分类网络即可。论文试验了两种网络：5 层的 ZF，16 层的 VGG-16。该图像中，额外添加一个 conv+relu 层，输出 51*39*256 维特征（feature）。

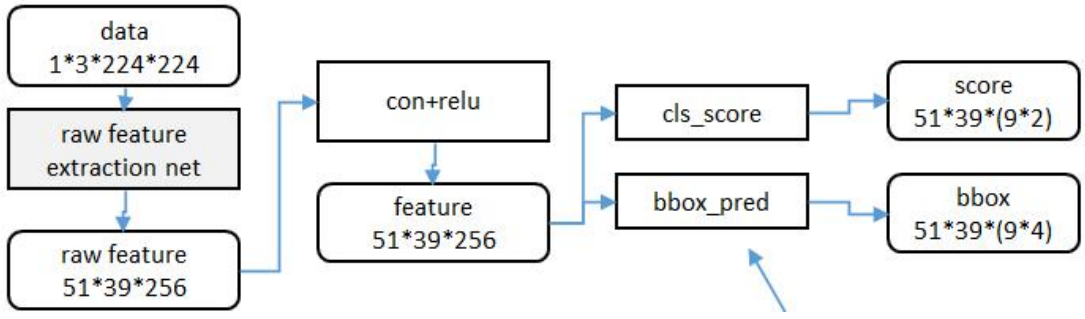


图 3.3.4 Faster RCNN 的基本流程图

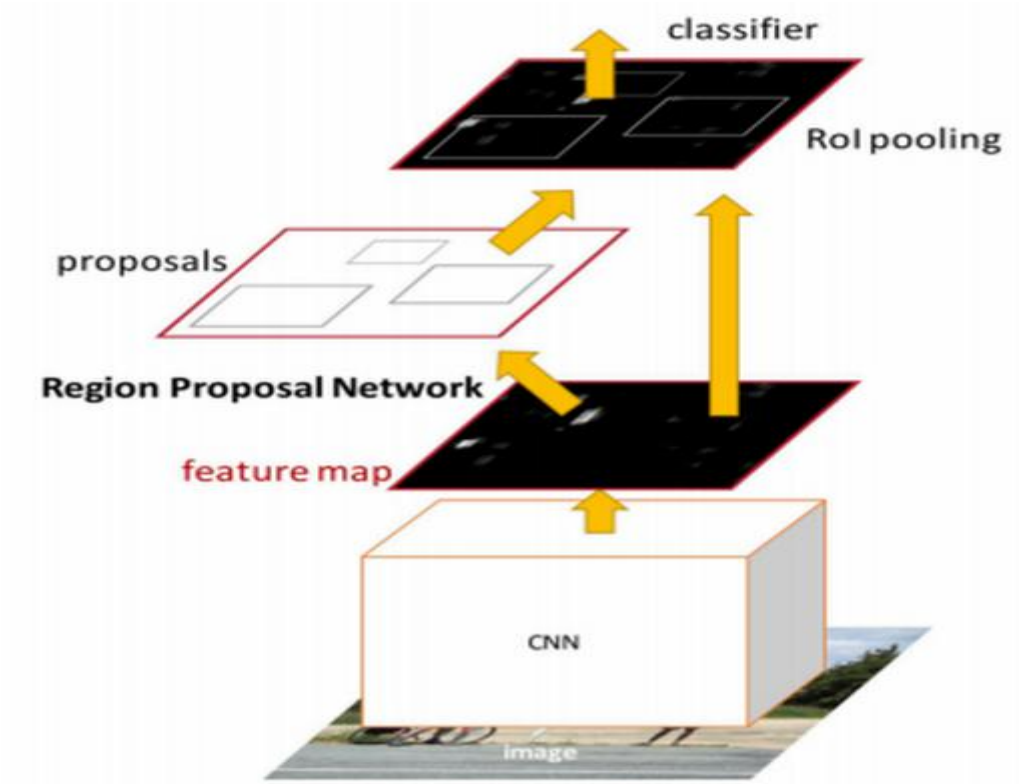
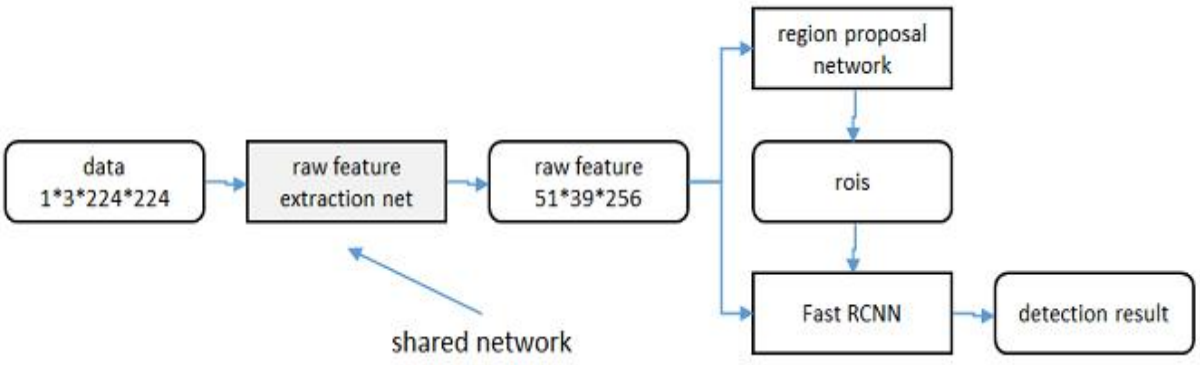


图 3.3.4 Faster R-CNN 架构图

区域生成网络（RPN）和 fast RCNN 都需要一个原始特征提取网络（下图灰色方框）。这个网络使用 ImageNet 的分类库得到初始参数 W_0，但要如何精调参数，使其同时满足两方的需求。论文中的训练过程如下：



a. 从 W0 开始，训练 RPN。用 RPN 提取训练集上的候选区域
b. 从 W0 开始，用候选区域训练 Fast RCNN，参数记为 W1
c. 从 W1 开始，训练 RPN…

具体操作时，仅执行两次迭代，并在训练时冻结了部分层。论文中的实验使用此方法。 如 Ross Girshick 在 ICCV 15 年的讲座 Training R-CNNs of various velocities 中所述，采用此方法没有什么根本原因，主要是因为“实现问题，以及截稿日期”。

3.3.5 Mask R-CNN

Mask R-CNN[32]可以准确的识别出目标的边缘，而不是简单的矩形框的样式，其思想是，对于每一个候选区域 Faster R-CNN 有两个输出，一个类别标签，一个矩形框坐标信息。这里我们加了第三个分支用于输出 object mask 即分割出物体。

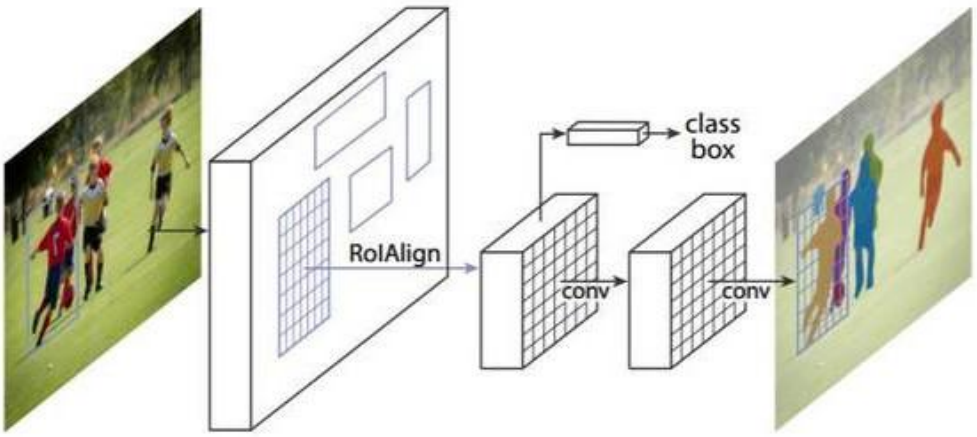


图 3.3.5 Mask R-CNN 处理基本流程图

3.4 声音论文

3.4.1 SoundNet

当前大数量的声音训练集匮乏，而图像相关的训练集则比较丰富。视频中包含图像、声音两个基本要素，而且绝大部分情况下，图像和声音所表达的内容是相同的（相互增强），SoundNet 基于这个基础，利用图像的特征来训练网络提取声音的特征。

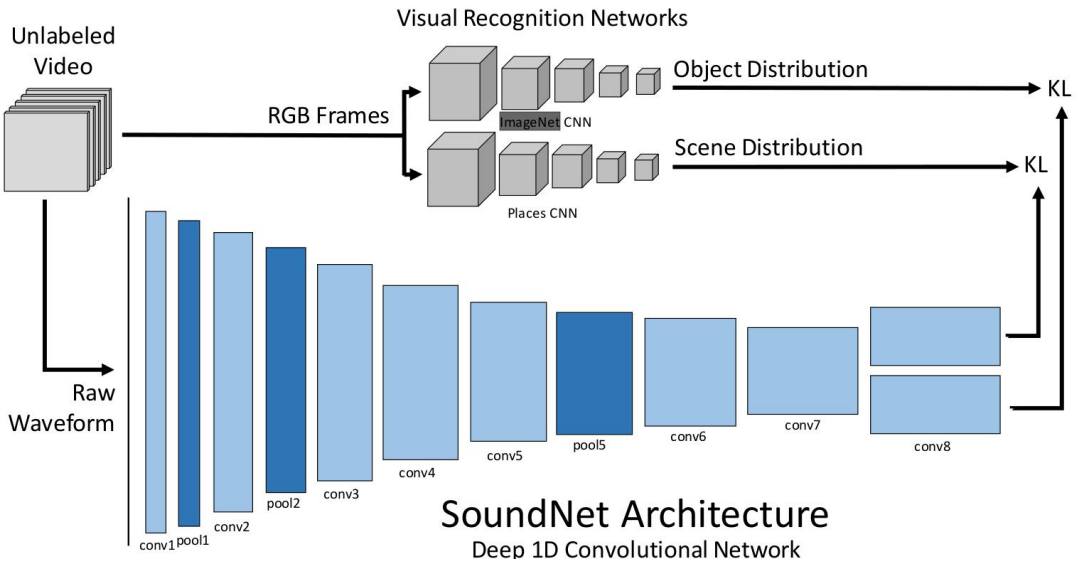


图 3.4.1 SoundNet 算法结构

SoundNet[33]算法的过程如下，使用 ImageNet 数据库训练一个识别目标物体的网络（成为 object CNN），使用 place 数据库训练一个识别场景的网络（scene CNN）；设计一个声音网络（类似 VGG 网络结构），最后的输出两个，一个为目标物体声音特征，一个为场景声音特征。在有丰富足够的视频数据集下，将同一个时刻的视频图像输入 objectCNN 和 sceneCNN，分别得到目标物体和场景，作为声音识别的目标，将音频输入声音网络，计算输出与目标的差值，反馈优化网络。

3.4.2 WaveNet

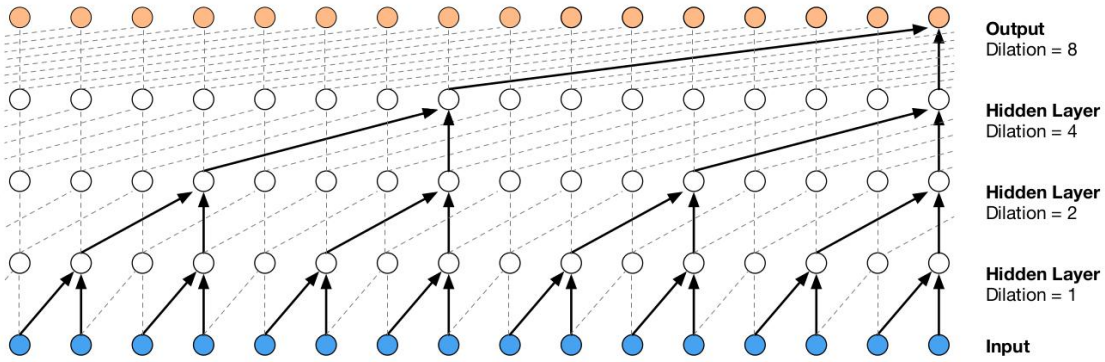


图 3.4.2 WaveNet 训练生成过程图

WaveNet[34]基本思想是 pixel RNN 从图像换到了音频，再加上 seq2seq training 的结果。

随后，[36]中 google 和 DeepMind 利用 WaveNet 的网络和训练方法，使用 Nsynth 音乐数据集[35]做音乐的合成，可以得到很好的听觉效果。

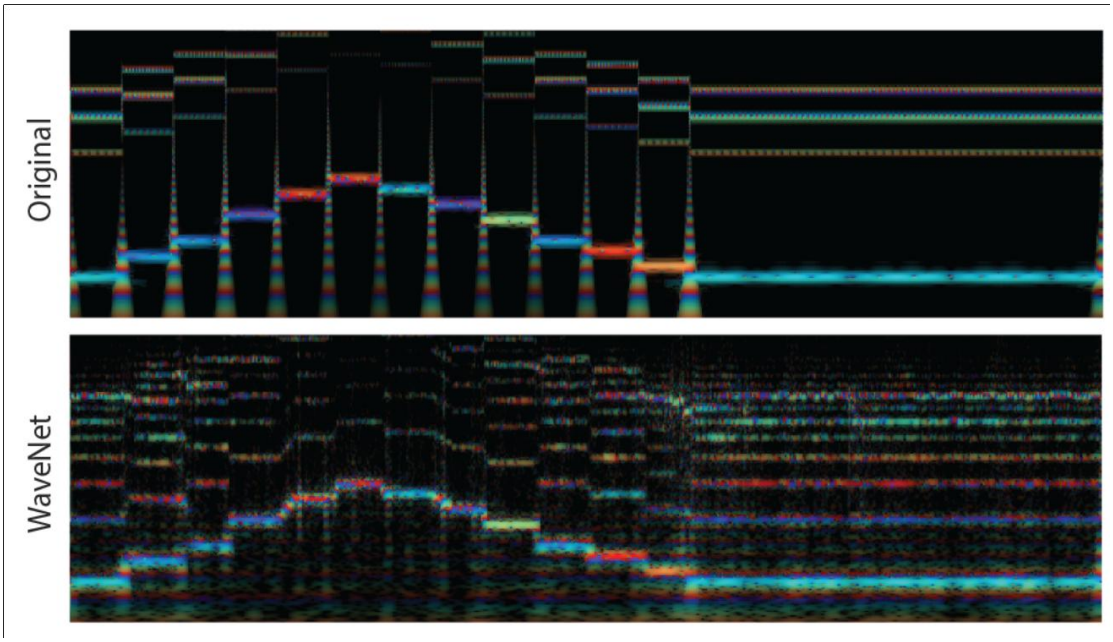


图 3.4.3 使用 WaveNet 和 Nsynth 合成音乐与原声音乐的特征图谱

按照论文的描述，使用 waveNet 合成的语音、音乐等都具有比较好的效果，但是需要有非常大的训练数据，而这个在很多情况下都是欠缺的。

4. 验证

本节会详细介绍我们在图像视频超清化、目标检测、水声识别等方面做的验证测试，并对结果做分析总结。

4.1 图像超清化验证过程的简介

传统的超清化算法研究主要在稀疏编码(spare coding)、邻域回归(Anchored Neighborhood Regressio)、贝叶斯(Bayes)等。而受技术限制，这些算法直接用在老视频的超清化中，效果不理想。

SRCNN 是第一次将超清化领域使用深度学习。不同于以往的利用特征表示，该方法采用端到端的方式，比较网络输出与目标图像之间的像素差值。网络的训练优化就是直接减少像素间的平方差值，从而达到提清的效果。SRCNN 是初次尝试，所以难免存在一些不足，后续工作逐渐做出改进。

VDSR 加深了网络层数达到 20 层，增大了网络，从而涵盖尽可能多的图像特征。该网络还使用了残差结构，极大的减少了训练难度和迭代周期，通过训练技巧的配合，有了进一步的超清化效果。

这两种方法所选用的 loss 都是图像间像素的平方差值，也就是提高了图像的 PSNR 指标。图像质量评估学中，PSNR 是常用的用来衡量图像质量的客观评价指标。但是 psnr 是对整个图像而言的，泛化能力较弱，而我们人类视觉会偏重在目标物体、边缘等。所以一些 PSNR 高的图像，视觉上会模糊，纹理细节等不突出。更多的图像质量评估，请参考附件[D.1]。

基于这样的情况，Perceptual-SR 调整网络的 loss，选用图像感知特性间的差值做 loss。也就是超清网络输出图像和目标图像，都再次做为训练好的做图像分类的 VGG 网络输入，

选用 VGG 网络某一层输出做图像感知特征，这两个特征差值做 loss。这种方式得到的超清化图像会有更好的视觉效果。

而 Perceptua-SRl 生成的图像还有一定的问题，主要表现在会有一些细小的问题，类似于弱化的棋盘纹。于是 SRGAN 考虑采用多种 loss 结合的方式：先用像素间的平方差做 loss 训练出一个比较好的效果，然后用感知特征 loss 和对抗网络网络 loss 相结合的方式做进一步的优化；EnhanceNet 更进一步，采用处理后的感知特征，使得生成的纹理细节更加真实。我们实际测试中发现，该方法对采用 Bilinear 下采样的图像超清化效果好，而对 Bicubic 的则效果不好。在处理老视频图像时，会将图像中的一些色块放大。

NTIRE2017 超分辨率挑战赛冠军方法（EDSR）在随后出现，该比赛也是以 PSNR 作为衡量指标，与我们期待的图像突出高频、产生纹理等不一致。在我们做了验证测试后，生成图像也确实是强调整体的接近，没有突出高频、问题等。

在基于图像处理的超清化效果有限后，我们分析老视频图像，发现图像画质差，包含的信息少（更多对老视频图像信息量的分析，参考附件[D.4]），所以验证利用多帧图像做超清化。虽然生成图像的 PSNR 有一点的提升，但是视觉上几乎没有差距。

我们在最后尝试视频图像的编码、场景数量、训练集数量、上采样倍数、不同 loss 等对图像的超清化的影响。从最终效果来看，虽然生成的图像有或多或少的不同，但是当以视频播放来看时，几乎感受不到差异，也没有特别惊艳的效果提升。

4.2 Waifu2x

Waifu2x 是比较早的利用深度学习做图像超清化的开源项目，提供提升 x2、x3、x4 等倍数的训练库，并且配有图像超清化的工具。该项目对动漫图像的超清化，效果比较好；但是对真实图像，特别是对对老视频图像超清化，则几乎没有改进。

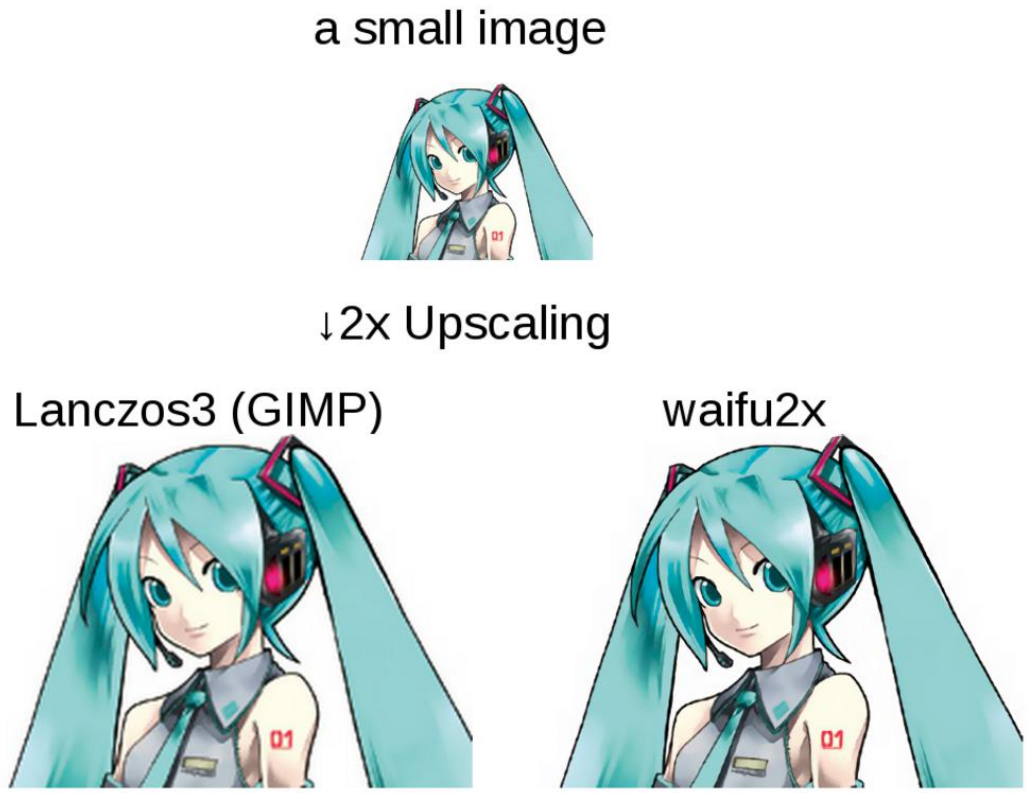


图 4.2.1，waifu2x 对动漫图像的超清化处理效果

图 4.2.2，waifu2x 对老视频图像的超清化处理前后效果，几乎没有差别

我们使用已有的 waifu2x 作为后续图像超清化的参考。

4.3 VDSR

4.3.1 论文效果复现

根据论文描述，使用 291 张图像生成 H5 格式的训练集（生成方法见附件），学习函数为 SGD，学习步长为 e-03，下采样方式为 bicubic，Y 单通道，网络采用高斯分布初始化；采用 caffe 框架，构建 VDSR 网络。训练过程中，测试集 Set5 的 PNSR 基本和论中的 PNSR 曲线一致，最终结果 PNSR 可以为 37.7。

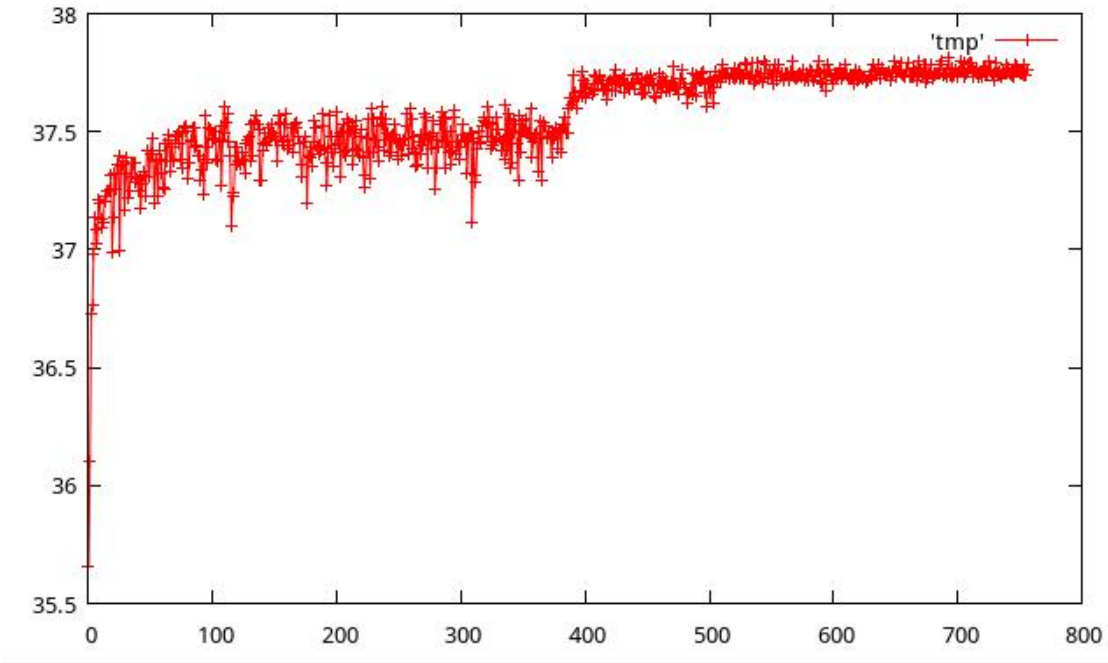


图 4.3.1 我们训练测试时的变化曲线，与论文中 psnr 变化一致

虽然复现了论文的效果，但是处理的老视频图像，几乎没有视觉效果的改进。我们知道，训练集的增多，会带来效果的一定提升，接下来，我们选用更多数量、更多类型的训练集做验证。

图 4.3.2 VDSR 处理老视频图像的效果

4.3.2 训练集

我们尝试通过增加训练集数量、特征等方式，来提升超清化效果：

- 291 张图像都包含比较多纹理，为模仿这些特征，在网上下载一批纹理特征丰富的图像，如动物皮毛、树叶、花蕊、建筑、任务、自然风景等，共计 108 张；
- 当时要处理的目标图主要是黑白老视频图像，我们在网上下载一批清晰度比较高的老黑白图像，共计 231 张；
- 在 google 的开源图像数据集 openimage 中，随机选择 1000 张图像；
- 90 年代《三大战役》视频，转换后的图像；
- 上面 3 和 4 两过程得到的图像，采用 photoshop 批量处理（主要是模糊化、调整色度等），得到低清图像。

上面前 4 种方式，采用 bicubic 下采样的方式得到低清图像，与原来高清图像做训练集对；仍然采用论文中描述的方式来做超清化的训练，而最终的效果，与使用 29 张图像的训练集相比，虽然 PSNR 有一定的提升，但是视觉上看，效果不明显。

第 5 种方式的图像集训练后，生成图像的亮度位置发生 变化，画面仍有模糊感。处理老视频图像，效果没有提升。

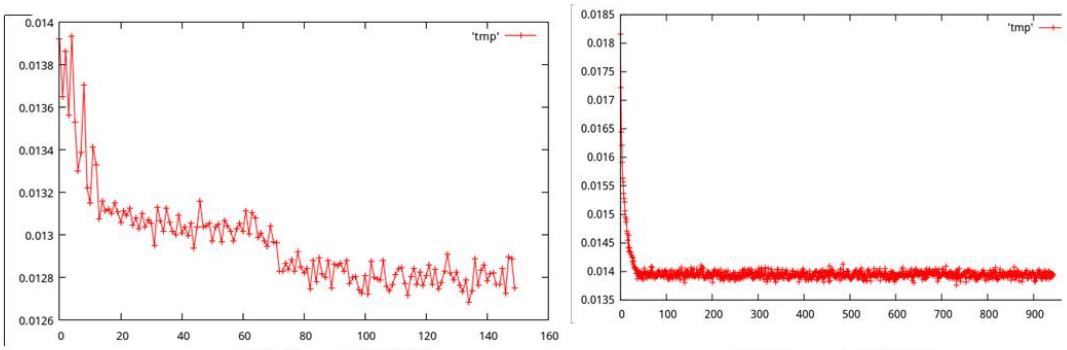
4.3.3 训练过程

在更多数据量、更多类型的数据集训练，没有效果提升的情况下，我们尝试做训练参数的调整，验证对效果的影响。

1. 不同的学习率：e=-2、e=-4。发现：高的学习率，开始时，loss 能快速下降，但是一段时间后，loss 不收敛（数值变大变小的波动）；而低的学习率下，loss 虽然不会产生波动，但是下降的慢。相依的训练技巧，是开始设置大一些的学习率，该学习率下，loss

能快速的下降，然后根据一定的规则，逐渐调小学习率，保证 loss 能持续的下降，尽量减少波动。

2. 优化函数：在深度学习中，使用最多的优化函数还是 SGD，参考[37-38]，从其描述和动态图上，看到 adadelta 方法似乎更优，使用该方法做测试。发现，使用 adadelta 方法的收敛速度没有 SGD 方法的快，效果也没有好。后续的论文，多采用 adam 算法，通过验证测试发现，adam 能得到更低的 loss 值
3. BatchSize：我们验证了 16、32、64 等三种不同的大小的 BatchSize，从最终的结果看，生成的图像没有差别。Google 的 2017-11 一篇论文[39]提出，通过增加训练过程中 batchSize，能够在训练集和测试集上取得类似学习率降低的表现。
4. 训练集构建的滑动窗口：在图像生成 H5 格式的训练集时，会做 dataAugment，除了常规的旋转、翻转之外，还会做图像的切换，如每隔 20 个像素，切割一个 64x64 大小的小图像。不同的像素间隔、不同的小图像大小，对应训练集扩大的倍数会不同。我们测试了几组不同的参数，发现最终结果没有差别。我们分析认为，这几组不同的参数下，岁仍然数据量有所不同，但是所包含的信息都已经充分，没有大的变化。
5. 不同的通道。我们分别验证对 Y 单通道和 RGB 三通道，采用 Y 单通道处理的图像，效果要好一些。黑白老视频图像，三个通道数值是相同的，所以可以只训练一个通道；单一通道变量少，所以也更好优化。



Adam函数的loss下降过程　　SGD函数的loss下降过程

图 4.3.3 SGD 和 ADAM 方法比较，adam 能有更好的收敛效果

4.3.4 总结分析

VDSR 是我们尝试的一个深度学习网络，我们复现了论文的效果，后续通过训练集、不同的优化，虽然在 pnsr 指标上有一定的提升，但是视觉效果上没有改进。我们熟悉了深度学习的基本训练过程，并基本掌握了一些调参，并总结了一些深度学习训练的经验。

1. DataAugmentation，很多情况下都需要做数据的扩展，用来充分挖掘出数据集中的特征信息。就图像处理而言，可做的 dataAugmentation 有翻转、旋转、随机截取、随机的扰动（添加噪声等）、输入数据的重洗（reshuffle）；
2. 初始化，采用预训练或好的初始化方法。验证中我们采用的初始化是符合高斯分布的初始化；在增加训练集时，采用之前训练过网络作为预训练的网络，后续的训练可以在此基础上变强；
3. 训练过程，我们验证中看到，学习率对结果有这比较大的影响，通过逐步降低学习率的方式，可以将效果微调优化（caffe 网络参数的配置调整，参考附件）；训练优化函数，优先采用 ADAM，其次考虑 SGD；
4. 我们实现了一个简单的脚本，可以图像化展示 loss 的变化、PSNR 指标的变化，更加具

体直观；

5. 采用 MSE(L2)做 loss 函数，即使做了各种努力尝试，得到的图像整体偏光滑，不能突出纹理、高频部分，对老视频图像的提升效果有限
6. 训练集的特征和测试集的特征要尽可能的相同

4.4 SRGAN

4.4.1 实现说明

我们以论文[15]为基础，参考开源项目[41-42]中的实现，使用 torch 作为框架，来实现 SRGAN 的图像超清化，其具体细节如下：

- 输入：RGB 三通道，采用和 VGG 相同的预处理(0-255 的取值范围，各个通道减去对应的均值)
- 网络：完全同 SRGAN 描述
- 残差块结构：图 3.1.2 中的“BN before Add”
- 训练集：使用 ImageNet2012 中所有长、宽均大于 96 的图像

4.4.2 梯度消失问题

在验证过程中，我们遇到了梯度消失问题和棋盘纹问题，很长时间才解决这两个问题，先介绍梯度消失的问题。

所谓梯度消失问题，就是随着训练迭代次数的增多，BP 到网络前面层的梯度逐渐变小甚至消失为 0，参考资料[43]深度神经网络为何很难训练，以公式推导的形式解释了梯度消失可能出现的原因，概括而言，就是从后往前，每一个神经层内的权重变小，就会导致网络前面神经层的反馈梯度变小。当出现梯度消失，但是对应的权重参数不会发生变化，也就出现了网络训练不动。

我们训练过程的梯度消失，表现为前面神经网络层的很多权重变的比较小（如 e-40），接近 0，而随着训练迭代次数的增加，更多的权重参数变小（如何查看神经网络中各层的权重数值，参考附件）。正常情况下，训练后的网络权重大小应该高斯分布：正数值和负数值接近，其绝对值部分为 e-3 或 e-4，越小的数值数量越小，偶尔有几个 e-8 或 e-9 的数值。网络层中引入正则化（BatchNormalization）、使用残差网络、长连接、ReLU 层，都可以缓解或解决梯度消失问题。我们的网络中这些方法都包含，可还是出现了梯度消失问题。
最终，我们对输入数据做均值归零化的预处理（输入数据的平均值为 0），用这样的数据训练，没有在出现梯度消失问题。

4.4.3 棋盘纹

参考论文的实现，先用 MSE-Loss 预训练网络，然后使用 VGG+GAN 的 loss 继续训练，迭代次数增多后，输出图像的具有带有斜的棋盘纹。

图 4.4.1 输出的具有棋盘纹的图像

这些棋盘纹会出现在图像中的各地地方，破坏了图像的视觉效果。Google 大脑的论文[44][45]分析认为，网络中的反卷积过程很容易产生“不均匀重叠”从而出现了棋盘纹。论文验证使用上采样的网络层代替反卷积层，可以消除棋盘效应。我们调整网络，将反卷积层改为上采样层，重复之前的训练过程，输出图像仍然具有“棋盘效应”。之后做了很多的测试，发现当生成训练集的下采样改为“**bilinear**”时，没有在出现棋盘纹。

为什么使用 bicubic 的下采样方式训练会出现棋盘效应，而采用 bilinear 的下采样方式则没有棋盘效应呢？附件中，有对不同下采样方式生成图像效果的具体描述，可以看到，torch 框架的 image 库，bicubic 处理后的图像，边缘差更大，图像的块更加明显，突出高清部分；而采用 bilinear 处理后的图像，整体模糊，边缘相差更光滑。这些不同的特征影响了网络的学习。

图 4.4.2 torch 采用 bicubic 下采样 4 倍 和 采用 bilinear 下采样 4 倍的效果图

这在一个侧面反应，深度学习超清化并不具有普适性。可能是一些特征的图像超清化，需要使用深度学习算法 A，而另外一些特征的图像超清化，需要使用深度学习算法 B；推测，也可能存在一些特征的图像，不适合采用深度学习做超清化。

4.4.4 效果

我们用 bilinear 下采样算法生成训练集，在图像输入前做均值归零化的预处理，解决了训练过程中遇到的问题，训练后，基本复现了论文图像超清化的效果。

原图　　论文中x4，vgg训练效果图

我们x4，mse训练后效果图　　我们x4，vgg训练后效果图

图 4.4.3 SRGAN 图像超清化的效果比较

但是处理老视频图像时，输出的图像存在如下的问题

- 没有可见的清晰度提升
- 图像颜色有轻微的变化
- 图像中随机出现比较大的污点，视频观看闪烁严重

图 4.4.4 srgan 训练库处理老视频图像的效果

采用 bilinear 的训练集不适合黑白老视频图像，所以在接下来，我们尝试不同方式的训练集，能否更适合黑白老视频图像，从而带来效果的提升

4.4.5 总结分析

- Torch 平台当有问题时，采用 print 的方法逐步定位到问题；注意 cpu 和 gpu 上数据的交换（data:cuda(), data:float()）。
- 使用 BatchNormalization、ReLU、残差结构，输入图像做均值归零化的预处理，可以很大程度上避免网络权重变 0、梯度消失。
- 采用上采样方式代替反卷积层，输入图像特征更加光滑，可以消除输出图像的“棋盘效应”。
- SRGAN 采用 16 个残差块（效果和训练代价之间的权衡），是一个更大的网络，可以学习“记住”更多的特征。采用 VGG+GAN 做 loss，既可以产生图像的高频部分，又可以让图像更加的真实，从而产生更好的效果。
- 采用‘bilinear’下采样方式生成的数据集，训练的网络不适合黑白老视频图像的超清化，是否有别的方式生成数据集能符合黑白老视频的图像特征呢。

4.5 EnhanceNet

在 SRGAN 的基础上，我们同样采用 Torch 框架实现 EnhanceNet，两者之间的不同主要在

如下如下几个方面

- EnhanceNet 使用 10 层残差块，SRGAN 使用 16 层残差块
- EnhanceNet 的长连接是 input 之后到 Output 之间，Srgan 的长连接是 Input 之后到上采样之前

4.5.1 效果比较

EnhanceNet 与 SRGAN 两者输出的图像很近，两者采用了非常相近的训练过程。EnhanceNet 处理老视频图像效果不理想；SRGAN 输出图像存在一定的颜色变化，而 EnhanceNet 输出图像则几乎没有颜色变化。

图 4.5.1　EnhanceNet 和 SRGAN 输出图像比较

4.5.2 总结分析

EnhanceNet 比 SRGAN 少了 6 层残差块，但是输出图像效果近似，可以认为 10 层残差块的网络，就有能力“学习记住”我们设置的图像特征。更大的网络（SRGAN）可以学习到更多的特征，从而有更多的效果提升；但是这些提升在视觉上不明显。这也就是我们前面提到的，网络的增大虽然可以带来效果的提升幅度是逐渐降低的，从而要在效果和实际开销之间取折中。

黑白老视频常见的分辨率是 352x288，使用 SRGAN 网络处理这么大小的一张图像时，因为网络太大，内存不足，需要将图像分成两部分分开处理然后合成一张；而使用 EnhanceNet 可以直接处理整张图像，在没有可见的效果变差情况下，可以显著的提高速度。

EnhanceNet 的长连接部分没有对图像做卷积操作，传递了输入图像的完整信息，这样尽可能保证了输出与输入的颜色相一致。这种方式也会存在一个问题，如果图像中存在稍微严重的损坏，也会将损坏部分直接传递到了网络的末端，导致输出图像的损坏也很明显，我们在图像损坏修复时，就遇到这样的情况。

4.6 EDSR

该团队开源了代码[46]，同样采用 torch 框架。使用相同的比赛数据集，我们训练 14 天，复现论文的效果。EDSR 处理图像的 pnsr 指标比较高，但是不能较好的生成纹理、突出高频部分。从下图，我们可以看到，EDSR 处理老视频图像的效果并没有变好。

图 4.6.1 EDSR 处理下采样图像，老视频图像的效果

4.6.1 Edsr 方法验证

EDSR 相比较之前的 SRGAN、EnhanceNet 的方法，处理除了采用比较大的网络外，还对网络结构、loss 等做了改动。因为 EDSR 的网络比较大，处理一张 352x288 的图像大概需要 2-3s，不能满足处理视频图像的需要，主要还是考虑小一些的网络。接下来，我们测试 EDSR 中的改动是否能带来超清化效果的提升

.Loss 使用 L1

相同点： EnhanceNet 的网络结构，戏剧视频压缩 2 倍得到的训练集，ADAM 学习函数，lr=e-04,Y 单通道均值归零化预处理，迭代 400 周期；

不同点： 方法 A 采用 L1-Loss，方法 B 采用 L2-Loss；

结果： 从结果中我们看到，方法 A 输出的图像显得更加的光滑，高频部分等越不明显；视觉效果上看，方法 A 的输出图像比方法 B 的输出图像差；

分析： 相比较 L2 方法，L1 是看整体的单个像素之间的直接差值，更加强调整体上差值的小，所以输出图像会更加光滑。

结论： 尽量不使用 L1 作为 loss。

.去掉 BatchNormalization

相同点： 近似 EnhanceNet 的网络结构，戏剧视频压缩两倍得到的训练集,ADAM 学习函数，Lr=e-04，Y 单通道均值归零化预处理，L2-Loss，迭代 400 周期

不同点： 方法 A 的网络没有 BN 层，方法 B 的网络中有 BN 层

结果： 方法 A 的迭代 200 个周期后，loss 值几乎不动，生成的图像还是比较模糊
结论： 网络结构中继续使用 BatchNormalization 层

.残差块中的缩放
相同点：近似 EnhanceNet 的网络结构，戏剧视频压缩两倍得到的训练集,ADAM 学习函数，Lr=e-04，Y 单通道均值归零化预处理，L2-Loss，迭代 400 周期
不同点：方法 A 在残差块中使用参数为 0.1 的缩放，方法 B 不使用
结果： 两者输出的效果没有可见的差异
结论： 网络的残差块结构中不使用缩放

.多种旋转翻转组合
相同点： EnhanceNet 的网络结构，戏剧视频压缩两倍得到的训练集,ADAM 学习函数，Lr=e-04，Y 单通道均值归零化预处理、且做旋转翻转等增加数据集，L2-Loss，迭代 400 周期
不同点：方法 A 采用源图像输入，网络输出就作为结果；方法 B 将图像做旋转、翻转等得到 8 种情况做输入，网络输出后做相应的相反旋转、翻转后组合成新的图像
结果： 视觉上图像的差距不大；客观指标上，方法 A 得到图像的 PSNR 更高
结论： 方法 B 增加了计算量且没有带来效果的提升，我们不采用

4.6.2 总结分析

EDSR 项目使用更大的网络，以及网络结构、Loss 函数的调整。而网络过大不适合我们的产品应用场景；其他调整，在我们的应用中也没有带来效果的改进。该方法主要是 PSNR 指标的增加，与我们希望的突出高频部分、更好的视觉效果不一致。

所以我们了解到了深度学习图像超清化的一些思想，而不在继续采用该方法。

4.7 多帧超清化

4.7.1 背景

从公开的深度学习超清化效果看，以 Set14 测试集为例，图像下采用 x2 后超清化，psnr 可以增加近 3.8Db，下采样 x4 后超清化，pnsr 只能增加 2.9DB。图像下采样 x4 后，所包含的信息量源少于下采样 x2 的图像，下采样 x4 后提清的结果，不仅没有下采样 x2 后提清的结果，甚至没有下采样 x2 不提清的效果。在我们的实际测试中也发现，输入图像的包含的信息量多，处理后的图像效果越好；而输入图像信息量少，处理后的图像效果就相对比较差。关于图像信息量多少的评估考虑，参考附件。

基于此，我们需要增加输入图像的信息量，因为视频中每帧图像的信息固定的，我们当然想到了利用前后帧的信息，增加图像的信息量。

4.7.2 验证测试

我们要测试，采用多帧输入的方式，是否确实能带来确实可用的超清化效果提升。我们

比较单帧、多帧两种方式做图像超清化的效果，具体描述如下：

相同点： 采用近似 EnhanceNet 的网络结构，ADAM 学习函数，Lr=e-04，RGB 三通道均值归零化预处理，L2-Loss，迭代 400 周期

不同点： 方法 A 完全采用 EnhanceNet 的网络结构，戏剧视频压缩两倍得到的训练集；
方法 B 的输入为连续三帧，然后两两结合，得到单一输入后使用 EnhanceNet，长连接采用中间帧，戏剧视频压缩两倍后使用 AVS 做运动估计得到训练集，输入是整张图像。

结果： 随机的选取两个压缩后的图像测试，比较客观指标，发现多帧输入处理，能有更高的 PSNR 指标；但是**主观上比较图像、视频，没有差别**。

表 4.7.2：单帧多帧处理图像后的 PSNR 指标

	up	单帧训练库	多帧训练库	EDSR训练库
图1	30.89	33.05	33.43	33.95
图2	34.88	37.12	39.43	39.09

4.7.3 总结分析

在计算机视觉领域里面，单帧和多帧的超清化是两个独立的方向了。单帧超清化主要是使得输出结果显得自然清晰，多帧算法才希望得到真实的细节。在我们测试中，利用视频中前后多帧，虽然能增加一定的信息量，能增加一定的指标提升，但是在主观效果上看，几乎没有改进。

从实现方便性与效果两方便综合考虑，我们的超清化不采用多帧的方式。

4.8 其他尝试

表 4.8.1，训练集和测试集采用相同、不同的下采样方式时，图像的超清化效果

训练集采样 Psnr 变化 测试集采样	Bilinear	Bilinear	Bicubic
Bilinear	3.85	3.35	1.55
Bicubic	-0.95	0.94	3.76
	Python 平台	Matlab 平台	Matlab 平台

4.8.1 尝试方法

从上面表格很明显的看出，如果训练集和测试集采用相同的下采样方式，验证指标高，图像的超清化效果比较好；而如果是不同的下采样方式，验证指标低，图像的超清化效果不好。所以，我们尝试采用不同的方式来生成训练集，试图找到更符合老视频图像特征的方法。

构造方式

1. 多种下采样方式结合：分析老视频图像，我们不能确认其特征更符合哪种下采样方式。

我们这样认为，每次获取训练数据对是，随机的选择一种下采样算法，这样训练数据就会包含各种特征，从而也可以包含老视频图像的特征。

2. 新旧对应视频构建：采用人工操作为主，软件辅助的方式，目前已经修复了一批经典的黑白老电影。使用新旧视频可以得到低清、高清的图像对，这样的图像数据完全就是老视频图像，特征相符合。但是这个构建的时候存在很大的问题，主要是新旧图像的像素无法对应一致。
3. 视频压缩方法：是在方法 2 不可操作的前提下，采用视频压缩的方法生成训练集。数字化视频都是采用一定的编码（时空压缩）封装到视频文件中，我们采用与黑白老视频相同的编码格式，降低视频的分辨率和采码率，压缩生成低清的视频。高清、低清视频所对应图像构成训练集。选用的高清视频是一些戏剧视频。
4. 相同场景的视频压缩训练集：采用同 3 相同的方式，选用的高清视频是修复版的高清“小兵张嘎”视频，而测试数据是损坏的旧版本“小兵张嘎”视频。这样训练集和测试集有相同的场景，可能也有相同的特征。

方式 2 构建训练集存在这样的问题：新老视频的获取方式不同，导致两个视频间的帧数不同，画面的大小不同，没有可循的规则让新老视频画面的像素相匹配。

还曾经有过如下的构思：

1. 借鉴 GAN、VGG 思路，先训练好一个深度学习网络，判断图像的好坏（高清图像得到高，低清、损坏图像得分低），使用这个网络做 loss 评价。考虑到 GAN 网络难训练，很多情况是随着训练迭代次数的增加，对抗网络的判别能力提升，而生成网络的生成结果却逐渐变差，所以预先训练好这个判别网络，从而减少了对生成网络的干扰。测试结果，生成图像结果变得更差。分析原因，预先训练判别网络时，输入图像的好、坏都比较明显，但是中间过程的图像好坏就会判断错误，也就是这个判别网络的判断不准确。
2. 训练集、测试集都做相同处理，这样尽量让训练集和测试集有相同的特征，训练后测试，是否会有好的结果呢？从前后的测试经验分析，不会。因为最终生成图像的结果，与输入原始图像的好坏有非常大的关系，而开始的处理，降低了输入图像的质量，因而输出效果可能会更差。

4.8.2 测试效果

1. SRGAN 的超清化能有比较好的表现，所以先采用 SRGAN 中的网络结构、参数、训练过程（除低清图像的获取方式外），训练后输出图像都带有或多或少的棋盘纹。
2. 数据集 3，采用 SRGAN 的网络结构、参数，类似 VDSR 上的训练过程：Y 单通道均值归零化处理、MSE 做 loss。训练后处理黑白老视频图像，相比较 VDSR 的结果，有一定效果提升，但是不明显。视频压缩时，不同的采码率也会有不同的效果： xxxx
3. 数据集 4，也按照测试 2，采用 SRGAN 的网络结构、参数，类似 VDSR 上的训练过程：Y 单通道均值归零化处理、MSE 做 loss。训练后处理损坏的旧版本“小兵张嘎”视频图像，效果没有数据集 3 得到的训练库的效果好。

4.8.3 总结分析

1. 采用视频压缩方法得到的训练集，与老视频图像的特征更加接近
2. 训练接和测试集场景相同并不能带来超清化效果的提升
3. 大部分特征的训练集，使用 SRGAN 中的 VGG-GAN-Loss 权重，都会产生棋盘效应

4.9 优化尝试

4.9.1 背景

前期的工作中，验证了不同训练集的获取、网络 Loss、网络结构，得出应该采用什么组合。

表 4.9.1 生成训练集的不同方法，采用视频压缩方式效果最好

方法	分析	结论
老图像，人工处理到高清图像	处理难度大，开销太大	不可行
修复过的视频，低清和高清两版本图像	1.两视频帧、像素很难匹配 2.有高清版本的视频，是七八十年代电影，可能与五六十年代电影有差别 3.这种电影少，训练集的数量和特征也就受限	不可行
图像下采样	目前已经测试过这些下采样算法，对老视频图像效果不好	不采用
图像压缩	图像压缩也是采用 MPEG 等压缩算法，但缺少分辨率的下降	不采用
图像模糊化	不考虑运动模糊 资料拍摄、数字化转录过程可能会存在模糊、平滑化等 只能使用非常轻微的模糊、平滑	不采用
图像调黑	像素值按照一定比重变黑，适应老电影偏暗的特征化 视频增强工具可以优化	不采用
视频压缩	视频使用的编码方式是确定的，如 MPEG 系列、H.26x 系列 数字化(VCD、DVD)->rm(vb)等转化，先用 mpeg-1 压缩，后转 rm 或 rmvb 格式，尽可能与目标视频的处理一致 设备原因导致拍摄视频不清晰不能覆盖	**采用**

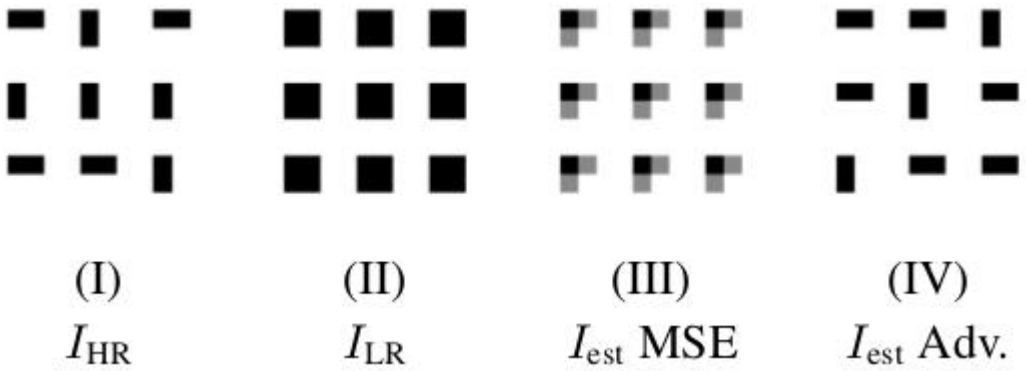


图 4.9.1 L2 与对抗网络 Loss 理论效果图

表 4.9.2 不同 loss 函数的选取

Loss	分析	结论
L1	优势：EDSR 项目采用；	不采用

	缺点：画面太过光滑，不能突出高频部分	
TV	优势：图像受噪声污染越大图像的总分量（TV）就越大 缺点：降低 tv-loss 减少图像的污染(噪声等)	不采用
FSIM	优势：FSIM 最接近主观评价； 缺点：训练集生成图像做测试，视觉好的图像，fsim 数值并没有高；且复杂难实现	不可行
L2	优势：使用广泛，对应 PSNR 指标的提升； 缺点：不能突出高频部分	采用
VGG	优势：可产生细节纹理、突出高频部分 缺点：会出现伪细节，计算量大，需要三通道数据	采用
GAN	优势：减缓人工 Loss 与视觉 loss 不一致的影响，填充缺失信息 缺点：网络发散难训练	采用

表 4.9.3 网络结构的选取方式

网络结构
X2：skipcon(Conv -Res_B16 -up2 -CBR -Conv), X4: skipcon(Conv -Res_B16 -up2 -CBR ***-up2 -CBR*** -Conv) CBR: Conv_BN_Relu Res_B: skip_con(CBR-CB) up2: UpSamplingNearest
X4 与 x2 不同点：***-up2 -CBR***

基于此，我们我们测试如下几点对超清化效果的影响，是否有效果的优化。

表 4.9.4 优化测试点

测试点	**对比 1**	**对比 2**
视频格式	Mpg 格式视频	MPG-RM 格式视频
视频 avs 处理	MPG 格式视频	**AVS 处理过**的 MPG 格式视频
视频压缩倍数	压缩 x2 倍	压缩 x4 倍
场景数量	3 个视频	8 个视频
样本量	每 20 帧取 1 个	每 1000 帧取 1 个

4.9.2 视频格式验证

基准数据：

3 个戏剧视频，压缩 2 倍为 mpg 格式(格式化工厂)，采码率 768k，ffmpeg 转图像，每 20 帧留取 1 个图像放入训练集，共 2w 张

测试点：

训练集 1，直接用 mpg 格式视频

训练集 2，mpg 格式视频转 RM 格式(realproducterPlus8.5，real8.0 格式，354K)

结论：不采用 RM 格式视频生成训练集

结果描述：

rm 训练集结果，出现了明显的颜色变化(蓝色)，主要在色块部分，变得突出；

mpg 效果稍好一点（即使图像都黑白化处理后）

分析：

rm 视频图像，有光滑模糊处理，导致颜色的变化；rm 视频也出现独立颜色的色块

rm 只是做进一步的时间上压缩，采用的编码与 mpg 视频一样，都是 h.264-mpeg 标准

测试 2 背景：

生成 rm 格式图像与 rm 老视频图像，非常相近，特别是 色块内都有小量竖着的小细纹。却因样色问题，导致没提升

测试：

格式转换时使用 386K 采码率转为 rm 格式；

使用 Y 单通道测试

效果：RM 格式视频训练集，效果没有提升

结论：进一步验证不使用 rm 格式视频来生成训练集

4.9.3 AVS 处理

基准数据：

3 个戏剧视频，压缩 2 倍为 mpg 格式(格式化工厂)，采码率 768k，转图像，每 20 帧留取 1 个图像放入训练集

测试点：

训练集 1：mpg 格式视频，直接用 ffmpeg 转图像

训练集 2：mpg 格式使用，使用 avs 脚本处理后，直接导出图像

结论： avs-x2 处理后的图像，没有提升

分析： avs 处理视频，改变了图像的通道表示方式，图像没有发生实质改变

4.9.4 SR 倍数

基准数据：

3 个戏剧视频，压缩为 mpg 格式(格式化工厂)，采码率 768k，ffmpeg 转图像，每 20 帧留取 1 个图像放入训练集，共 2w 张

测试点：

训练集 1，视频压缩 2 倍

训练集 2，视频压缩 4 倍

结论：X4 伪细节过多，导致视觉效果变差

结果：

后图像增加了细节，但是色块边缘等也放大，影响视觉效果；

视频看，伪细节导致前景与背景分离；

即使采用 deblock 后的图像，仍然存在很明显的各种伪细节；

4.9.5 场景多少

基准数据：

视频压缩 2 倍为 mpg 格式(格式化工厂)，采码率 768k，转图像，每 20 帧留取 1 个图像放入训练集

测试点：

训练集 1，3 个戏剧视频，2w 张

训练集 2，8 个视频(3 个戏剧，3 个 80 年代，2 个 90 年代），6w 张

结论：训练集中场景的增加，能增强训练结果

结果:

场景增多的训练库，生成的图像更加光滑，相应的也就丢掉了很多高频部分；

增加 VGG 的权重(如 0.1)，或者先用 mse 预训练，在用 vgg 优化，都会产生斜的棋盘纹

分析:采用 MSE+0.01VGG+GAN 做 loss，其目标接近 Mse-Loss，图像整体光滑，均化高频信息

更少的场景数量测试

基准数据： 3 个戏剧视频，压缩 2 倍为 mpg 格式(格式化工厂)，采码率 768k，转图像

测试点

样本集 1：每 20 帧取一张图像

样本集 2： 每 2w 帧取一张图像，22 张

结论：样本更少的训练集，处理效果好

效果：

样本 1 的结果比较光滑，高频部分不突出

样本 2 增加细节，视频来看，效果更好。也会产生振铃等伪细节

分析：

样本多训练的结果是趋近全局最优（输出图像与目标图像整体接近），图像表现光滑特点

样本少训练的网络只是一个局部最优，图像没有过于光滑，又能表现出一定的高频、细节

4.9.6 样本量大小

基准数据：

8 个视频(3 个戏剧，3 个 80 年代，2 个 90 年代），压缩 2 倍为 mpg 格式(格式化工厂)，采码率 768k，ffmpeg 转图像，取图像放入训练集

测试点

训练集 1：每 20 帧取 1 个图像，共 6w 张

训练集 2：每 1000 帧取 1 个图像，共 1.2k 张

结论： 样本量多的训练集，生成图像的视觉效果并没有直观的提升

结果：

训练集 1 的图像光滑

训练集 2 的图像能突出一些高频部分，有细微的振铃

分析:

视频同一场景会使用很多帧，1/20 后和 1/1000 的数据量，包含的场景并没有少太多

预处理时做旋转、翻转等，在一定程度上补充特征

4.9.7 棋盘纹

基准数据

8 个戏剧视频，压缩 2 倍为 mpg 格式(格式化工厂)，采码率 768k，转图像；x2 网络

测试点

样本集 1：VGG(9,36*0.1)*0.2 + MSE + GAN 做 loss

样本集 2： VGG(9)*0.1+MSE+0.0001*GAN 做 loss

结论：样本 1 无棋盘纹；两者视觉效果相差不大

效果：样本 1 的结果有棋盘纹变化了一些的伪细节

分析：GAN 能学习产生更真实图像，在一定程度上减少棋盘纹的出现

4.9.8 局部最优效果

采用斯大林 x4 训练集，mse 做 loss 时，当训练迭代 300-400 个周期时，处理老视频图像效果最好：能突出高频部分、去掉了影响视觉效果的伪细节、画面没有太光滑。继续训练，处理图像的效果而变差：画面变光滑。上面的 4.9.5 节第二个测试中，采用更小的样本时，得到的图像反而有更好的结果。

这两种情况，都是在局部最优时，图像超清化效果更好，可能是因为 loss 目标与我们期待目标不严格一致（MSE 会使图像光滑，VGG 可能会引入棋盘纹等伪细节），这种情况下，网络训练迭代次数越多，引入的错误信息可能就更多，效果就变差。而局部最优时，已经学习到了好的效果，有没有引入过度的错误信息，从而效果更好。

4.9.9 结论

1. RM 视频格式、avs 处理得到的训练集，对处理效果没有提升，不考虑
2. 训练集中场景的增多能增强网络的训练预期；而样本数量的增加，对预期的增强，效果不明显
3. 网络的局部最优情况下，处理后的图像效果更好

我们还尝试采用深度学习的方法最遥感图像目标识别与水声目标识别，在另外的几篇文档中我们做了总结，此处就不再赘述。

5. 经验

5.1 深度学习项目过程

参考资料[46] 是吴恩达机器学习工程实践中的经验总结，非常实用且独一无二的一本书，短小精悍但干货十足。我们借鉴该书籍的内容与思想，总结深度学习项目开展应该采取的过程。

5.1.1 目标与指标

首先，我们要有非常明确的目标，并且有准确的评价指标。比如遥感船只识别，目标就是在遥感图像上准确的识别出舰船、准确的识别出舰船的类型，使用识别率和误报率加权数值作为评价指标。图像超清化，其目标是有更好的视觉效果，涉及到了图像质量学，客观的评价指标和主观评价指标还有比较大的差异，如果采用主观评价，又受评价人、环境等因素影响，不能前后一致，且浪费时间。

像这种情况，我们采用的方式如下：训练时统计观察 loss，当 loss 下降比较比较平缓时，其训练效果就不会再有大的提升；之后，采用相同的输入，比较输出结果，主观评估好坏。也因为相近的图像，主观评价相差不大，所以对于深度学习，我们要选用视觉效果确实明显比较好的图像。

5.1.2 训练集

在深度学习中，训练集起到相当重要的作用，既要保证训练集的特征与实际应用场景中的特征相同，又要保证有足够多的特征。

对于图像的超清化，因为可以获取图像的方式非常多，所以训练集数量的要求比较容易满足；而特征一致性的要求，难度比较大，我们先后尝试多种方式生成训练集，测试验证，采用视频压缩方式得到训练集图像对，特征与黑白老视频图像比较一致。

对于水声目标识别的应用场景，水下环境本身就复杂多变，噪声也多，对训练集数量的要求也就更高。而不同目标在水下的声音或涉密，或很少或几乎没有采集，这样数据量少，并且也没有找到好的声音数据集扩展的方式。在没有训练集，或训练集不足的情况下，就不能考虑采用深度学习的方式。

5.1.4 网络构建

在深度学习广泛讨论的短短几年内，深度学习的各种网络模型先后出现。目前我们主要采用卷积神经网络结构（CNN）：

- 图像超清化，多使用残差结构(Residual Block)、长连接（skip connection）；
- 图像生成，如损坏部分的生成，多采用 Encoder-Decoder 的模型
- 生成对抗模型中的对抗网络，多采用类似 VGG 结构的方式
- 网络大小的选择，在没有可直接参考的资料项目基础上，需要实验尝试，选择大小合适的网络。从经验来看，在 Tesla 的 GPU（12G 内存）上，当网络中的残差块超过 10 个（chan=64）时，处理 352x288 的图像时，就可能会出现内存不足的问题。

5.1.5 训练过程

此处，我们假设训练集特征与应用场景的特征一致。在训练过程中，可以先验证某种网络结构是否满足要求，一般采用如下的顺序：

1. 小网络、小训练集，非常快速验证这种方式是否可行
2. 大网络、小训练集，较快速的验证在确定训练集的情况下，网络是否可讯、能否达到预期
3. 大网络、大训练集，从可以让 loss 降低的学习率开始，并在训练过程中不断的降低学

习率，让 loss 不断降低，到评价指标不再提升、或提升很少为止。从经验可知，多采用 ADAM 学习函数，学习步长多从 lr=e-03 或 lr=e-04 开始。

到目前为止，我们能选用的网络结构也就这么几种，在各种网络效果仍然不理想的情况下，需要尝试其他的 loss 函数。

5.1.6 何时结束

- 如果是比赛项目，则需要将评价指标尽可能的提高，增加数据集、DataAugmentation，降低学习率等等，直到评价指标不再有增加为止；
- 如果我们的评价指标准确、要求明确，训练时只要指标没有达到预定目标，且输出指标还逐渐有提升，则应继续训练；若投入很大的训练时间、成本，而指标提升不理想，则训练应该结束，这时需要优化网络、优化 loss 函数
- 如果我们的评价指标没有那么明确，或者一些提升在效果上不明显时，在验证寻找网络、loss 的训练时，可以考虑今早的结束训练。仍以图像的超清化为例，大部分情况下，开始训练时，loss 下降很快，输出图像的效果有很明显的改变，在迭代一定的周期（如戏剧视频生成的训练集，300 个周期）后，输出的图像虽然仍然有改变，但是都只是非常细微的改变，主观评价很难给出好坏。在比如 SRGAN 遇到的棋盘纹，期望随着训练迭代次数的增多，能消除或减少棋盘纹，结果却有变得更加明显的趋势。
- 我们还应采取 Early stop， 避免过拟合。

5.1.7 可视化

可视化是以图像形式展示训练过程的趋势、训练结果、网络的中间特征等，从而有更加直观形象的掌握整个训练。

- 趋势：网络训练 loss、输出评价指标的趋势图，因为训练过程迭代的次数、周期比较多，采用取平均或间隔采样的方式取一定量数值展示。附件提供了脚本，可以较便捷的展示趋势图
- 训练结果：在我们的超清化、图像生成等应用上使用较多，会训练迭代一定的步骤后，保存输入图像、输出图像、目标图像，直观的感受深度学习网络能改进多少，和目标还有多大的差距。
- 网络中间特性：多用在网络的调试优化上，为了看网络内的参数是什么特征，中间某层网络处理的数据是什么样子。我们做过这样的可视化，以直方图形式展示网络内各种权重大小的分布情况（代码见附件），输入某图像后，在 VGG 网络的中间某一层的输出图像。

5.2 问题总结与相应解决方案

梯度消失问题：使用 BatchNormalization、ReLU、残差结构，输入图像做均值归零化的预处理，可以很大程度上避免网络权重变 0、梯度消失。

棋盘纹问题：采用上采样方式代替反卷积层，输入图像特征更加光滑，可以消除输出图像的“棋盘效应”；VGG-GAN 做 loss 时，增加 GAN 的权重，也可在一定程度上避免“棋盘效应”。

5.3 训练技巧

训练集

务必保证有大量、高质量并且带有干净标签的数据，没有如此的数据，学习是不可能的。训练集的使用过程中最好采取如下处理。

- 预处理：0 均值化和 1 方差化
- 数据的扩展（DataAugmentation）：充分利用数据的特征
- 输入顺序的重洗(reshuffle)
- BatchSize：2 的指数倍，多采用 32、64 或 128

网络

- 结构：残差块、长连接的使用
- 网络层：使用 BN、ReLU 等层
- 核的大小：采用 3x3 即可，计算量少，而且与大（如 5x5）等没有差别
- Filter 大小：2 的指数倍，多用 64
- 网络大小：产业应用多采用 10-16 个残差块
- 初始化与预训练:影响很大，在没有预训练情况下，尽量采用 Xavier 初始化或 Kaiming（MSRA）初始化

Loss

- Loss 的设计要合理，能满足目标
- 训练过程中观察变化与波动
- 验证集上的 loss 作为 early stop 的依据

训练参数

- 学习函数：首选 adam，其次选用 SGD
- 学习率：用一个一般的学习率开始，然后逐渐的减小它；adam 可用 0.001 或 e-04，SGD 多用 0.1 开始；一般学习率降到 e-06(adam)、e-04(sgd)时，学习就会很慢，没有必要再下降了。
- 如果是迁移学习或者微调，最好采用比较小的学习率

6. 总结

6.1 深度学习理解

很多观点认为，深度学习是对数据的记忆，当然有观点认为记忆不容易，也有观点认为记忆非常容易[49]，而也有观点认为，深度神经网络的性能并非来自“记忆”， 而是源于在有限数据上学习简单的、切合的可用假设[48]。

深度学习需要大量的训练数据：是用大数据集，对深度学习网络做‘题海战’，强化训练出网络路径，在一个节点（网络层上的权重）上有多少数据量会流到下一层。而如果‘题海’中存在冲突，比如一些数据要增加这个节点的权重，而一些数据要减少这个节点的权重，那就要看这相冲突双方数据量的大小，因为训练过程是一个“强化”过程，哪方的数据量大，带来的强度就会越大，所以就就更偏向相应的效果。这样就解释了训练过程中的很多结果，在数据量小的情况下，各个特征几乎是相同的，处理训练集也会有比较好的效果；而训练数

据的增加，可能就会存在一定的冲突，再处理训练集中的数据，效果就没有小训练集时的效果好。

深度学习在小噪声场景表现突出：深度学习在分类任务中，具有非常好的成绩。大部分观点认为，这样的网络中，前面的层是提取数据的特征，后面很少的几层是做特征的分类。我们人工提取的特征，可能没有深度学习提取的特征好（特别是在图像目标分类），所以之前一直没有大的突破。但是深度学习的分类对噪声、干扰的敏感度太高，如图像中添加一些噪点，会将“大熊猫”识别成“猿猴”；语音识别中，安静环境下的识别准确率可以在 97%以上，但是环境中的噪声的增加，识别度就会迅速下降。

深度学习需要有着非常明确的目标：也就是 loss 指标，这样网络就会学习着符合这个目标。比如，围棋的评价指标就明确固定，通过完全自学习训练出来的 AlphaZero 也都可以有非常好的结果；DeepMind 采用相似的方式，训练网络玩“暴风游戏”，却还没有好的结果，可能是因为不能有明确的评估每一步的结果。图像的超清化，我们的目标是更好的视觉效果，而这个目标还没有好的评估指标（虽然图像质量评估学发展了很多年）。

深度学习是在大数据的统计，让统计的结果符合我们的明确目标[50]，没有逻辑推理，与人脑的思维相差很大[51]。当然，很多“模拟大脑”的项目也都没有取得成功，而深度学习深度学习并不是在“模拟大脑”，却取得了很大的成绩。

深度学习适合大数据量、小噪声、指标明确的任务。

6.2 项目总结

项目[52]采用 PSNR/SSIM 两个指标，对比了传统方法和深度学习方法对图像超清化的效果， 虽然只能在一定程度上说明算法的优劣，效果可能不准，但从各方面考虑，我们还是比较认可其结果，深度学习的方法比采用传统方法做图像的超清化能有更大的效果提升。深度学习做图像超清化的效果，受如下几方面影响：

1. 图像本身的清晰程度、信息量的多少
2. 训练集与实际应用场景的相似程度
3. 深度学习算法的影响（网络、loss、训练过程等）

通过各种测试验证，我们采用视频压缩的方式生成训练集、残差网络结构、MSE+VGG+GAN 组合 loss 等方式，超清化图像结果最佳，但是受老视频图像本身不清晰、信息量少的影响，处理后的图像没有惊艳的效果提升。因为对深度学习的盲目信任，我们对效果期望又比较高，所以花费很长时间做效果的提升。通过这些实验，我们认为采用深度学习虽然最近很热门，但是在一些应用场景还是有局限，比如做老视频的超清化，很难有理想的效果。我们对项目的开展也有一些总结：

1. 当有新的算法或项目时，我们应该先复现论文或项目中描述的效果，一是为了证明算法确实可行；二是可以节省时间，实现所需要时间少，且能根据效果确切分析是否需要后续的改进；三是掌握算法实现的经验，为算法的改进、效果的提升积累经验。
2. 验证过程时，在没有比较切实可行的理论下，可能要尝试各种方法，特别是深度学习，本身就是偏重工程实践，因为理论相匮乏，大部分是有了实际的测试结果后，再分析各种方法可能会有什么实际效果影响。要分析某个方法的影响，就应该保证其他的因素都不变，只有这一个因素发生变化，控制好变化的量，多组不同测试，从而了解每个因素的影响。这种对比结果可行，而且整体上也节省了时间，避免了可能出现的重复验证、各种因素爆炸组合条件的验证。
3. 深度学习特别是考虑图像质量的应用领域上，在遇到问题时，通过训练集、训练迭代次

数的增加，往往很难有好的效果提升，应该尝试其他不同的方式，如训练集特征的变化、输入图像预处理方式、网络 loss 等。也就是训练集、迭代次数等的增加，不是解决问题的方法，只是提高效果的一种方式。

附件

A. 软件安装

本章节主要是介绍在 Ubuntu16.04 环境下，安装配置深度学习框架相关的软件，我们根据实际操作与网上资料结合，方便以后的使用。

A.1 Nvidia 相关软件安装

Nvidia 相关软件主要包括 GPU 的驱动、CUDA 和 CUDNN 的安装，简单来说，驱动是为了显卡工作，CUDA 是英伟达为 GPU 应用到深度学习上的解决方案，CUDNN 是对这些解决方案的一些加速。

首先，去英伟达的官网上下载显卡驱动、CUDA、CUDNN 相关的软件包，下载的时候，要选择正确的 GPU 型号和操作系统平台。我们的 gpu 是:，操作系统平台是 Ubuntu16.04，我们已经下载好了软件包，分别是：

驱动：*NVIDIA-Linux-x86_64- 375.20.run*

cuda： cuda-repo-ubuntu1604-9-0-local_9.0.176-1_amd64.deb

cudnn：cudnn-8.0-linux-x64-v5.1.tgz

#安装依赖库：

apt-get ins freeglut3-dev build-essential libx11-dev libxmu-dev libgl1-mesa-dev libglu1-mesa libglu1-mesa-dev libxi-dev

#添加显卡可执行权限，并安装

chmode +x NVIDIA-Linux-x86_64- 375.20.run; ./NVIDIA-Linux-x86_64- 375.20.run

#cuda 安装[57]

sudo dpkg -i cuda-repo-ubuntu1604-9-0-local_9.0.176-1_amd64.deb

sudo apt-key add /var/cuda-repo-<version>/7fa2af80.pub

sudo apt-get update

sudo apt-get install cuda

CUDNN 安装

cd cudnn-path

sudo cp lib/ /usr/local/cuda/lib64/*

sudo cp cudnn.h /usr/local/cuda/include

sudo ln -sf /usr/local/lib/libcudnn.so. 5.1.5 /usr/local/lib/libcudnn.so.4

sudo ln -sf /usr/local/lib/libcudnn.so.5 /usr/local/lib/libcudnn.so

A.2 Caffe 平台安装

在正确安装了 cuda、cudnn 等工具之后，安装 caffe 相对就是比较容易的事情了。参考[53]下载好 caffe 的安装包，解压后得到 caffe 目录

cd caffe
cp Makefile.config.example Makefile.config
vim Makefile.config 如果想用 cudnn，则 USE_CUDNN:=1
make all
make test

当前的路径就是 caffe 的安装路径，使用 bin 下的 caffe 命令便可以做相关执行了

A.3 Torch 平台安装

Torch 自带安装脚本，比较方便[54]

git clone https://github.com/torch/distro.git ~/torch --recursive
cd ~/torch; bash install-deps;
./install.sh

在实际使用过程中，还会用到很多的依赖包，使用 ‘luarocks’命令即可。例如我们处理图像会使用 image 包，则如下安装

luarocks install image

A.4 Tensroflow 平台安装

安装 GPU 版本的 TensorFlow[55], *Sudo pip install –upgrade https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow-0.8.0-cp27-none-linux_x86_64.whl*

B 工具

B.1 视频图像转换

在 Linux 平台上，我们使用 FFMPEG 工具做图像和视频的转换，ubuntu 下，使用’aptitude install ffmpeg’命令即可安装。因为视频图像编码的问题，转为 PNG 格式图像、mkv 或 MP4 格式视频，视觉效果非常好。Ffmpeg 命令可以支持很多的功能，在我们的项目中，比较多的使用操作主要包括：

ffmpege -i video.rm –f image2 f-%06d.png #整个视频转图像
ffmpege –t 350 –I video.rm –f image2 –ss 300 f-%05d.png #300s 到 350s 之间转
ffmpege –f image2 -i f-%05d.png –r 25 out.mkv #设定视频帧率

两个视频合并到一起左右显示，用于对比

*ffmpeg –i left.mkv -vf "[in] scale=iw:ih, pad=2*iw:ih [left]; movie=right.mkv, scale=2*iw:2*ih [right]; [left][right] overlay=main_w/2:0[out]" -b:v 9000k merge_y.mp4*

添加声音：

ffmpeg -y -i out.avi -i out.wav -vcodec copy -acodec copy gen.avi

B.2 平台训练库转换

目前使用的方式，主要是将 caffe 的训练集转为 torch 平台。VGG 官方项目网站上提供 VGG16 和 VGG19 的训练库，是基于 Caffe 平台的，而我们运行 SRGAN 算法时，主要是在 torch 平台上。以 VGG19 为例，采用这样的 torch 脚本可以做两个平台只见的转换：

```
require 'loadcaffe'
model = loadcaffe.load('VGG_ILSVRC_19_layers_deploy.prototxt', 'VGG_ILSVRC_19_layers.caffemodel', 'nn')
torch.save('VGG19.t7', model)
```

B.3 图像目标标签工具

用于快速便捷的标注图像中的目标，生成目标训练集。我们选用开源项目 BBox-Label-Tool[56]，该工具操作简单，可视化程度高，但是存在如下几个问题：1. 图像加载到固定大小的区域，不能调节；2. 记录的区域位置是以坐标百分比的形式，与后续采用像素坐标识别的深度学习网络不兼容；3. 图像类型是按照目录来分类的，且必须是数字的形式，如果一个图片中有多种类型目标物体，就不能正确的标记分类。

我们对代码做了调整，解决了上面的问题，从而适应我们的需求。

B.4 训练过程可视化工具

训练过程中 loss 的可视化，先采用脚本过滤全部的 loss 值，因为数值太多，直接全部显示，会震荡厉害，很难直观的看出变化趋势。我们根据实际情况，一定数量的迭代次数取一个平均值，图像显示这些平均值。

```
run_train.sh > info
grep –o “err: [0-9.]*” info | awk ‘{print $NF}’> tmp
end=35
stride=100
function avg() {
  for i in `seq 0 $end`
  do
    startidx=`echo "$i*${stride} + 1" | bc -l`
    endidx=`echo "($i+1)*${stride}" | bc -l`
    sum=`sed -n "${startidx},${endidx}p" loss | awk '{sum+=$1}END{print sum}'`
    echo "$sum"/${stride} | bc -l
  done
}
```

avg

C.资料说明

我们将整个项目用的资料汇总在同一个文件下，下面对这些资料说明

```
data      使用到的相关数据
  | image，标准或非标准的图像库
  | video，训练、验证等的视频
  | sound，声音、音乐、水声等
src       相关的源代码
  | vdsr    caffe 版本
  | srgan   torch 版本
  | enhanceNet   torch 版本
  | videoSr   torch 版本
  | EDSR      torch 版本
  | others    其他的项目
  | script    使用的脚本，README 或脚本开头做功能介绍
result    深度学习测试验证的结果
  | train-models 得到的训练库
  | image         输出的图像结果
  | video         视频结果
paper     整个过程中参考的文章
  | sr
  | sound
  | ..
```

D. 理论

D.1 图像质量学评估学

图像质量的评估，分为主观评价和客观评价，通俗来讲主观评价，就是人类自己来评价图像的好坏，客观评价就是按照一定的算法，给出图像质量的一个指标数值。

很显然，主观评价的结果非常的可信，但是会存在这样的问题：结果会因人、因时间而异，存在一定的波动性，而且很难给出具体的数值（特别是在图像质量比较相近的情况时），且评估开销非常大。

客观质量评估，虽然可以给出具体的指标数值，但是在一些情况下，这些指标会不准确，与人类视觉有很大的出入，特别是在没有目标图像的情况下。目前使用比较多的图像质量客观评价指标是 PSNR 何 SMI，目前仍然没有评价准确地客观评价指标。

DXO： 为了更加全面的掌握图像质量评估，研究了比较权威摄像机镜头评估标准 DXO，该标准考虑如下几点：ISO 灵敏度、SNR、动态范围、色阶范围、色彩灵敏度、信噪比等几方面，在标准测试图上拍照，评估效果。这与评估已有的图像质量不同，后续我们没有采用。

FSIM：出现时间比较晚，声称与主观图像质量评估非常的相近，而我们实际测试了几张图像，发现该客观评估指标还是很大的出入，后续也没有在采用。

图 D.1.1 不同方式下生成的图像，SRCNN 和 Ours(Lfeat)相比，虽然有较高的 PSNR(客观评价指标)，但是主观评价却没有后者好

D.2 图像清晰度

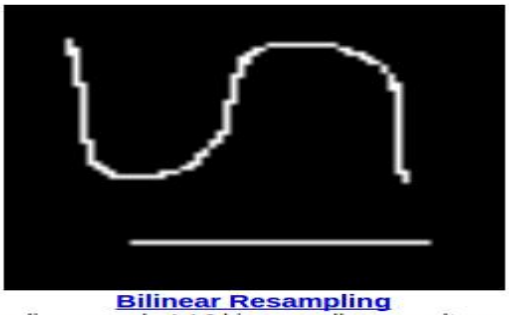
不考虑损坏情况下，图像清晰度是**分辨率**和**反差**(锐度)的综合表现。反差：鲜锐度、明锐度，是摄影镜头鲜明地再现摄景物中间层次、暗部层次、低反差影纹细节、微弱亮度对比和微妙色彩变化的能力。反差高的镜头，所成影像轮廓鲜明、边缘锐利、反差正常、层次丰富、纹理细腻、影调明朗、质感强烈、色彩过渡柔合、彩色还原真实、自然。

图 D.2.1 高分辨率与高反差的图像更加清晰

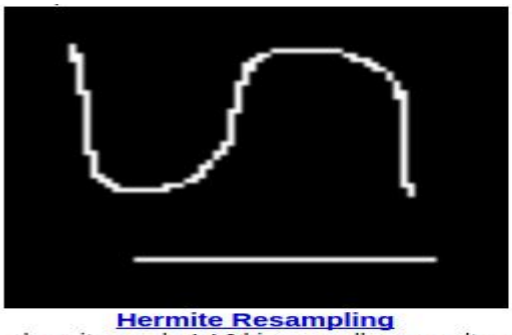
D.3 采样方式

图像采样方式使用比较广泛，此处我们列出图像经过几种不同采样算法处理后的效果图[58]，以便图采样有更加直观的了解，没有采样算法的理论分析。

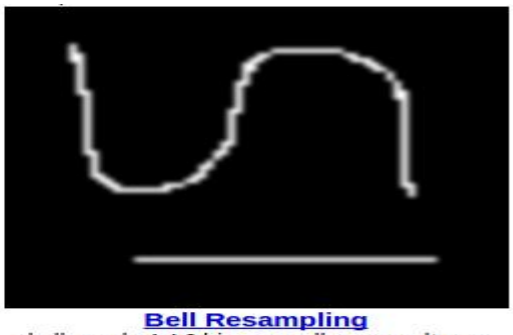
Nearest Neighbor Resamplingn
prescale 4 4 0 binarysmall.pan result.pan
Temps: 0.004s



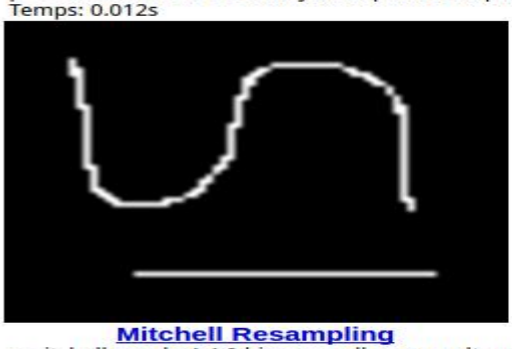
Bilinear Resampling
plinearrescale 4 4 0 binarysmall.pan result.pan
Temps: 0.019s



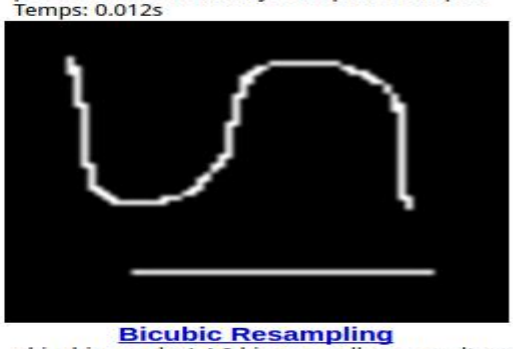
Hermite Resampling
phermiterescale 4 4 0 binarysmall.pan result.pan
Temps: 0.012s



Bell Resampling
pbellrescale 4 4 0 binarysmall.pan result.pan
Temps: 0.012s



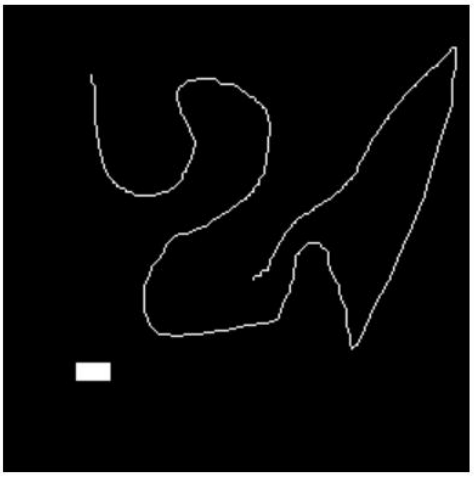
Mitchell Resampling
pmitchellrescale 4 4 0 binarysmall.pan result.pan
Temps: 0.043s



Bicubic Resampling
pbicubicrescale 4 4 0 binarysmall.pan result.pan
time: 0.022s

Lanczos Resampling
planczosrescale 4 4 0 binarysmall.pan result.pan
Temps: 0.302s

图 D.3.1 不同上采样算法下的图像效果图



Nearest Neighbor Resampling

prescale 0.4 0.4 0 binary.pan result.pan
Temps: 0.006s

Bilinear Resampling

plinearrescale 0.4 0.4 0 binary.pan result.pan
Temps: 0.006s

Hermite Resampling

phermiterescale 0.4 0.4 0 binary.pan result.pan
Temps: 0.009s

Bell Resampling

pbellrescale 0.4 0.4 0 binary.pan result.pan
Temps: 0.015s

Mitchell Resampling

pmitchellrescale 0.4 0.4 0 binary.pan result.pan
Temps: 0.037s

Bicubic Resampling

pbicubicrescale 0.4 0.4 0 binary.pan result.pan
Temps: 0.006s

Lanczos Resampling

planczosrescale 0.4 0.4 0 binary.pan result.pan
Temps: 0.423s

图 D.3.2 不同下采样算法处理效果图

D.4 图像信息量

我们分析高清图像与老视频图像的直方图、色系、图像边缘、信息熵、压缩比等方式，试图找到能正确评价老视频图像信息量的方法。

直方图、色系等表示图像中颜色的多少与分布情况，一般情况下，高清图像（即使是下采样后的图像）的颜色范围比较广(0-255)，分布也比较均衡，而老视频图像的颜色分布则没有那么广，集中在中间偏小（偏暗）。之前电视等视频的颜色取值在(16-235)之前，可能有一定影响。但老视频图像中，直方图、色系的问题不是广泛存在的，一些质量比较好的图像，其直方图、色系等与高清图像的比较接近。

我们分析图像的信息熵，压缩比，每个视频开始处随机选取 3 张图像取值的平均，得到如下表的结果，图像压缩比更具有参考价值，而且老视频图像压缩比更低。

表 D.4.1 随机选取视频图像的压缩比、信息熵

	无损压缩比 (应占 bits)/(无损压缩后 bits)	信息熵∑p(xi)log(2,p(xi))
白毛女	5.35	6.39
刘胡兰	5.80	6.52
慈母曲	10.40	6.28
游击队	8.38	4.23
小兵张嘎	3.88	6.80
地雷战	4.14	6.79
戏剧-搜书院	2.08	7.36
戏剧-李后主	1.86	7.16
斯大林	3.07	6.15

对不同压缩比的图像，我们测试，发现：

- 较低压缩比，如小兵张嘎（3.156）SR 处理后，清晰度有改善
- 居中压缩比，刘胡兰中某图像（5.66），SR 处理后，图像有变化，但清晰度不是很明显
- 较高压缩比，刘胡兰中某图像（17.88），SR 处理后，图像没有看到变化
- 图像压缩比可以标识图像中包含的信息量的多少，信息量少的图像 SR 处理效果不明显

D.5 水声目标识别简介

水下目标识别主要是采用水声识别，其他的信号在水介质中传播都会发生严重的衰减，此处我们主要讨论采用被动声呐识别舰船目标。目标舰船声音我们主要考虑稳态信号，包括螺旋桨转动噪声、机械噪声、水动力噪声等，从网上的一些资料推断，声呐兵识别时主要根据螺旋桨转动噪声做识别。

目标舰船有各种型号、各种国家，所涉及的分类众多，受复杂海洋环境(水温、盐度、密度、深度、洋流、表面及海洋生物噪声、气泡)的影响，接收到的声音相比原来的声音发生了各种畸变，混杂的各种背景噪声、混响等。

传统方法做声音识别，多会转为梅尔频谱或梅尔频率倒谱系数处理，识别效果有限。但

是受数据集的影响，采用深度学习做水声目标的识别也无法进行。

D.6 其他

. 病态问题

病态问题是指输出结果对输入数据非常敏感的问题，如果输入数据有微小误差，则引起问题解的相对误差很大，那么也就可以称这个问题为病态问题。

. 凸优化与非凸优化

简单的测试一个集合是不是凸的，只要任意取集合中的俩个点并连线，如果说连线段完全被包含在此集合中，那么这个集合就是凸集，如下图的左侧。

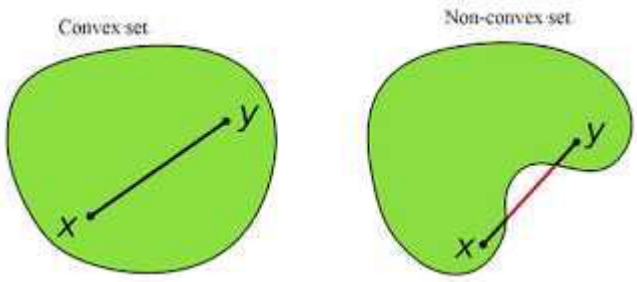


图 D. 6. 1 凸集合与非凸集合

凸集合内的问题优化属于凸优化，其处理相对简单，有个非常重要的定理，即任何局部最优解即为全局最优解。而非凸优化问题被认为是非常难求解的，因为可行域集合可能存在无数个局部最优点，通常求解全局最优的算法复杂度是指数级的（NP 难）。在深度学习、机器学习等领域，我们遇到的问题狠多是非凸优化问题，得到的解可能只是局部最优。在局部最优里，我们也通过各种技巧，找个一个最好的解。

[1] Zouxy, Deep Learning（深度学习）学习笔记整理系列，http://blog.csdn.net/zouxy09/article/details/8775360

[2] cs231,http://cs231n.github.io/convolutional-networks/

[3] http://blog.csdn.net/jacke121/article/details/54973802，深度学习框架比较

[4] 看的“深”看的“清”， https://zhuanlan.zhihu.com/p/26496124

[5]BPExample, https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

[6] Soheil B, 1511.06435, Comparative Study of Deep Learning Software Frameworks

[7] 主流深度学习开源框架对比，http://blog.csdn.net/app_12062011/article/details/54691945

[8] 深度 | 主流深度学习框架对比：看你最适合哪一款？http://mp.weixin.qq.com/s?__biz=MzA3MzI4MjgzMw==&mid=2650719118&idx=2&sn=fad8b7cad70cc6a227f88ae07a89db66#rd

[9] Top Deep Learning Projects, https://github.com/hunkim/DeepLearningStars

[10] 深度学习中 Batch Normalization 为什么效果好？https://www.zhihu.com/question/38102762

[11] 聊一聊深度学习的 weight initialization，https://zhuanlan.zhihu.com/p/25110150

[12] Dong C, Loy C C, He K, et al. Image super-resolution using deep convolutional networks[J]. IEEE transactions on pattern analysis and machine intelligence, 2016, 38(2): 295-307.

[13] Kim J, Kwon Lee J, Mu Lee K. Accurate image super-resolution using very deep

convolutional networks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 1646-1654.
[14] Johnson J, Alahi A, Fei-Fei L. Perceptual losses for real-time style transfer and super-resolution[C]//European Conference on Computer Vision. Springer International Publishing, 2016: 694-711.
[15] Ledig C, Theis L, Huszár F, et al. Photo-realistic single image super-resolution using a generative adversarial network[J]. arXiv preprint arXiv:1609.04802, 2016.
[16] Dahl R, Norouzi M, Shlens J. Pixel Recursive Super Resolution[J]. arXiv preprint arXiv:1702.00783, 2017.
[17] Sajjadi M S M, Schölkopf B, Hirsch M. EnhanceNet: Single Image Super-Resolutionthrough Automated Texture Synthesis[J]. 2016.
[18] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, Kyoung Mu Lee, Enhanced Deep Residual Networks for Single Image Super-Resolution, 2017, http://cv.snu.ac.kr/publication/conf/2017/EDSR_fixed.pdf.
[19] Armin Kappeler, Seunghwan Yoo, Qiqin Dai, and Aggelos K. Katsaggelos, Video Super-Resolution With Convolutional Neural Networks, IEEE TRANSACTIONS ON COMPUTATIONAL IMAGING, VOL. 2, NO. 2, JUNE 2016
[20] Xin Tao,Hongyun Gao,Renjie Liao,Jue Wang ,Jiaya Jia, Detail-revealing Deep Video Super-resolution, ICCV2017.
[21] Osama Makansi, Eddy Ilg, and Thomas Brox, End-to-End Learning of Video Super-Resolution with Motion Compensation, arXiv:1707.00471
[22] Waifu2x, https://github.com/nagadomi/waifu2x.
[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations (ICLR), 2015.
[24] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.
[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems (NIPS), pages 2672–2680, 2014.
[26] DeepLearning 回顾，https://www.cnblogs.com/52machinelearning/p/5821591.html
[27]残差网络 ResNet 笔记, http://www.jianshu.com/p/917f71b06499
[28] region CNN
[29] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” in European Conference on Computer Vision (ECCV), 2014.
[30] R. Girshick, “Fast R-CNN,” in IEEE International Conference on Computer Vision (ICCV), 2015.
[31] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.
[32] Kaiming He,Georgia Gkioxari,Piotr Doll ar,Ross Girshick, Mask R-CNN,ICCV,2017.
[33] Yusuf Aytar,Carl Vondrick ,Antonio Torralba, SoundNet: Learning Sound Representations from Unlabeled Video,2016.
[34] avdnoord, sedielem, WAVE NET: A GENERATIVE MODEL FOR RAW AUDIO, arXiv:1609.03499
[35] The NSynth Dataset, https://magenta.tensorflow.org/datasets/nsynth

[36] Jesse Engel,Cinjon Resnick, etc. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders, arXiv: 1704.01279.
[37] cs231 Gradient, http://cs231n.github.io/neural-networks-3
[38] 各种优化方法总结比较, http://blog.csdn.net/luo123n/article/details/48239963
[39] Samuel L. Smith, Pieter-Jan Kindermans, Quoc V. Le, Don't Decay the Learning Rate, Increase the Batch Size, arXiv:1711.00489
[40] Must Know Tips/Tricks in Deep Neural Networks, http://lamda.nju.edu.cn/weixs/project/CNNTricks/CNNTricks.html
[41] SR,https://github.com/leehomyc/Photo-Realistic-Super-Resoluton
[42] fast-neural-style, https://github.com/jcjohnson/fast-neural-style
[43]深度神经网络为何很难训练，http://www.jianshu.com/p/917f71b06499
[44] Augustus Odena, Deconvolution and Checkerboard Artifacts,2016
[45] 谷歌大脑：缩放 CNN 消除“棋盘效应”,http://www.sohu.com/a/118024734_473283
[46] edsr, https://github.com/LimBee/NTIRE2017
[47] Andrew NG, Machine Learning Yearning V0.5
[48] David Krueger1, Nicolas Ballas1, etc. DEEP NETS DON’T LEARN VIA MEMORIZATION, Workshop track - ICLR 2017
[49] Chiyuan Zhang, Samy Bengio, etc. Understanding deep learning requires rethinking generalization, arXiv: 1611.03530
[50] 深度学习：暴力计算与记忆，https://zhuanlan.zhihu.com/p/27210635
[51] 浅析 Hinton 最近提出的 Capsule 计划，https://zhuanlan.zhihu.com/p/29435406
[52] sr-benchmark, https://github.com/huangzehao/Super-Resolution.Benckmark
[53] caffe install in Ubuntu, http://caffe.berkeleyvision.org/installation.html
[54] torch get start, http://torch.ch/docs/getting-started.html
[55] tensorflow，http://wiki.jikexueyuan.com/project/tensorflow-zh/get_started/os_setup.html
[56] label box, https://github.com/puzzledqs/BBox-Label-Tool
[57] cuda, https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1604&target_type=deblocal
[58] RESCALING, https://clouard.users.greyc.fr/Pantheon/experiments/rescaling/index-en.html