

**《多媒体技术》课程项目**

项 目 题 目：图像风格迁移

所 在 学 院：计算机学院

项 目 组 长：张展通

小 组 成 员：吴伙强 庞文豪

目录

[**一、引言** 1](#_Toc74509089)

[**二、国内外研究现状** 1](#_Toc74509090)

[**三、模型和算法** 3](#_Toc74509091)

[3.1 VGG19预训练模型 3](#_Toc74509092)

[3.2 卷积神经网络 5](#_Toc74509093)

[3.3 池化 6](#_Toc74509094)

[3.4 图像风格迁移的过程 6](#_Toc74509095)

[**四、实验结果分析** 12](#_Toc74509096)

[4.1 实验一 14](#_Toc74509097)

[4.2 实验二 15](#_Toc74509098)

[4.3 实验三 16](#_Toc74509099)

[**五、结论** 18](#_Toc74509100)

[**六、参考文献** 18](#_Toc74509101)

# 一、引言

绘画艺术从文明的开始就一直伴随着人类，而绘画图片从计算机科学的角度可以抽象为内容和风格的结合，而神经风格转换就可以实现这样的技术。神经风格转换(NST)是深度学习中最有趣的技术之一，它合并两个图像，即内容图像 C（contentimage）和样式图像 S（styleimage），生成图像 G（generatedimage）。生成的图像 G 将图像 C 的内容与图像 S 的样式组合在一起。

图像中的内容和风格是可以分裂的，这使得在保留所需要语义内容的同时可以改变图像风格。其中一项突破性研究是Gatys等人的研究，他建议使用CNN将一张图片的内容分离出来，并将它们与另一张图片或艺术品的风格结合起来，创造出令人印象深刻的新艺术形象。随后，采用算法对随机图像进行优化，使其最小化随机图像与一张图像的内容和风格相似。

本项目就是采用卷积神经网络（CNN）和VGG19模型，采用tensorflow配合VGG网络。实现将输入图像经过过滤器后，能够提取特征图，然后进行图像风格迁移。

# 二、国内外研究现状

在2015年前，大多数以前的风格转移算法基于非参数的纹理合成方法。例如，Efors 和 Freeman 等人通过合并从输入图像中获取的补丁来计算输出纹理图像。Hertzman 等人使用图像类比将纹理从已经样式化的图像转移到目标图像上。Ashikhmin 从一个样本图像开始，生成一个任意大小的新图像，其外观与原始图像相似。尽管这些算法取得了不错的结果，但它们都受到相同的基础限制，它们仅使用目标图像的低级图像特征来进行纹理传输，因此得到的图像风格迁移效果并不是特别理想。

随着深度学习的兴起，Gaty等人提出一种基于卷积神经网络的高效参数纹理建模方法，通过与训练VGG模型提取图像的抽象特征表示，构造Gram矩阵作为图像的风格表征。图像风格迁移现有研究工作主要分为基于CNN的图像风格迁移和基于GAN的图像风格迁移两类方法。

Gaty等人最早发现卷积神经网络可以从任意图像中分离地提取图像的内容特征表示和风格特征表示，他们使用预训练VGG模型作为特征提取器和构造图像特征的Gram矩阵作为风格表示，利用图像迭代的方式直接优化初始噪声图像的像素，生成具有原内容和新风格的风格化图像。但是这种方式存在着速度缓慢和耗费内存的问题。

Johnson 和 Ulyanov等人提出了快速风格迁移方法，即基于模型迭代的风格迁移方法，但是这种方法在扩展到其他风格时的时间成本太高。因此，Chen等人提出一种在单个模型生成多种风格的方法，Zhang等人在生成网络种引入一个新的互匹配层，构建了一个多风格生成网络。

Li等人发现批量归一化中的统计量包括不同域的特征，通过调整匹配图像特征在通道方向上的均值和方差也能实现风格迁移。Ulyanov等人研究表明在快速风格迁移网络中使用实例归一化层（Instance Normalization ,IN）能够加快网络的收敛速度并降低风格损失。Dumoulin等人提出条件实例归一化层（Conditional Instance Normalization ,CIN），将网络中所有的卷积参数在多种风格之间共享，能够将输入内容图像转化为不同的风格。

在2014年，Goodfellow等人提出了一种独具一格的网络模型，生成对抗网络（GAN），该模型包含一个生成式网络G和一个判别式网络D，G和D的巧妙对抗设计，使该模型拥有强大的数据生成能力。Sanakoyeu等人在GAN中引入一种风格感知的内容损失，能够学习同一类艺术风格而不仅限于一种风格中的一个实例。Kotovenko等人在对抗网络中设计了一个内容转换模块，在具有相似内容信息的内容和风格图像之间的风格迁移过程中，学习如何改变内容的细节。

# 三、模型和算法

### 3.1 VGG19预训练模型

我们使用VGG19预训练模型，如图3-1所示。

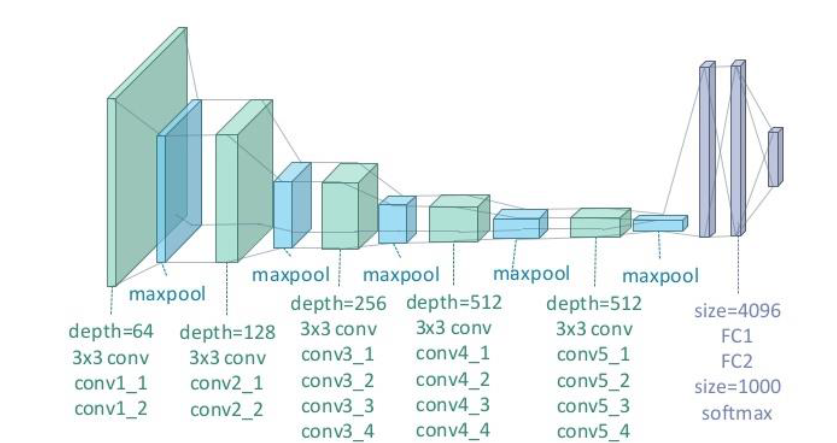


图3-1 VGG19预训练模型

VGG19预训练模型由16层卷积和ReLU非线性组成，由5个池化层分隔并以3个全连接层结束。VGG19每个卷积块包含不同数量的过滤器。第一个块有64个带池化步骤的过滤器，第二个块包含128个带池化步骤的过滤器，第三块包含256个带池化步骤的过滤器，第四块包含512个带池化步骤的过滤器，第5个块包含512个带池化步骤的过滤器，第 6 块包含全连接层。

**VGG19简介：**

1.VGG-19网络中的19代表的含义：19个隐藏层（16个卷积层和3个全连接层）

2.VGG-19网络结构很规整，没有那么多的超参数，专注于构建简单的网络，都是几个卷积层后面跟一个可以压缩图像大小的池化层。

即：全部使用3\*3的小型卷积核和2\*2的最大池化层。

3.优点：VGGNet的结构非常简洁，整个网络都使用了同样大小的卷积核尺寸（3x3）和最大池化尺寸（2x2）。几个小滤波器（3x3）卷积层的组合比一个大滤波器（5x5或7x7）卷积层好：验证了通过不断加深网络结构可以提升性能。

缺点：.VGG耗费更多计算资源，并且使用了更多的参数（这里不是3x3卷积的导致的结果），导致更多的内存占用（140M）。

### 3.2卷积神经网络

我们使用卷积神经网络，其结构如图3-2所示。：

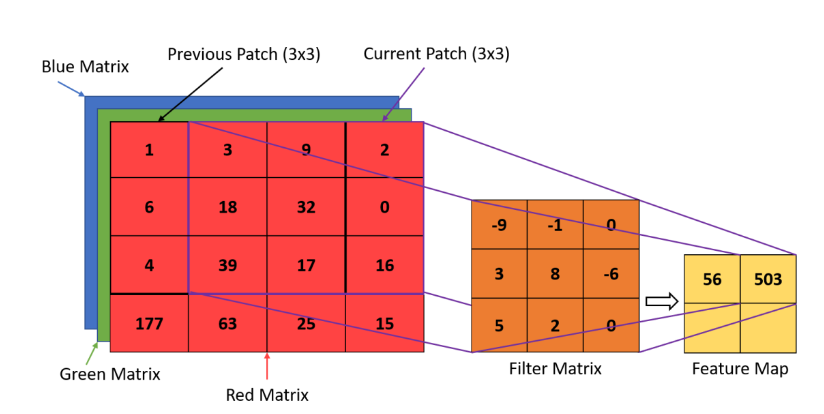


图3-2 卷积神经网络

在CNN的情况下，卷积的主要目的是从输入图像中提取特征。 卷积通过使用输入数据的小过滤器识别图像特征来保留像素之间的关系。每张图像都可以看作是一个像素值矩阵，VGG网络在每个卷积块中使用不同数量的过滤器，每个过滤器是一个3 x 3权重窗口。通过用过滤器扫描输入图像，我们得到一个特征图，它描述了整个输入图像中给定特征的存在。我们从每个卷积层得到的结果是一个3Dmatrix NwidthX NheightX Nconvolution而Nwidth 是图像宽度, Nheight 是图像高度，Nconvolution 是层中过滤器的数量。我们将滤波器矩阵在原始图像（在每个 RGB 矩阵上）滑动一个像素，称为步幅，对于每个位置，我们计算2个矩阵（输入图像和滤波器矩阵）之间的乘法和将乘法结果相加得到最终的整数，它构成了特征图的单个元素。

### 3.3池化

池化步骤降低了每个特征图的维数，但保留了最重要的信息，这一步可以是不同的类型：Max、Average、Sum等。在Max pooling的情况下，我们定义一个2 x 2的窗口并取从该窗口内修正特征图中的最大元素，然后将2 x 2窗口滑动2个单元格，这称为步幅，因为我们一次滑动2个单元格（步幅=2），每个卷积块中的特征图将减少一半。除了取最大的元素，我们还可以取平均值（Average Pooling）或该窗口中所有元素的总和（Sum Pooling）。对于图像合成，我们发现用平均池化代替最大池化操作会产生更吸引人的结果。

### 3.4图像风格迁移的过程

图像风格迁移的过程如图3-3所示，具体步骤如下：

1. 将内容和样式图像传递到VGG19网络中，以获得它们的特征表示矩阵。

2. 从其特征表示中计算样式图像Gram矩阵。

3. 初始化由随机噪声生成的图像X。

4. 通过生成的图像抛出VGG19网络得到其特征表示矩阵。

5. 从图像的特征表示中计算生成图像的Gram矩阵。

6. 计算内容损失:计算内容和生成的特征表示矩阵之间的均方误差。

7. 计算风格损失:计算风格和生成的Gram矩阵之间的均方误差。

8. 计算总损失:将内容损失和样式损失相加。

9. 使用优化算法计算梯度。

10. 根据梯度更新生成的图像像素。

11. 返回第4步。

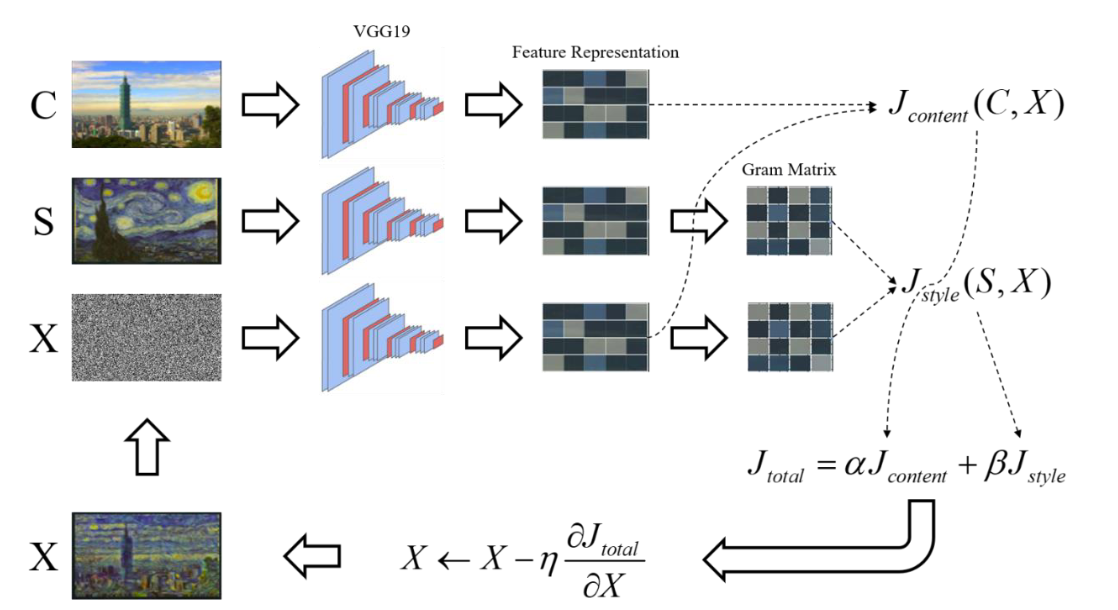


图3-3 图像风格迁移的过程

我们通过组合来自网络的许多不同层，可以捕获原始样式图像更精细的纹理和更大的元素，风格迁移网络的结构如图3-4所示。

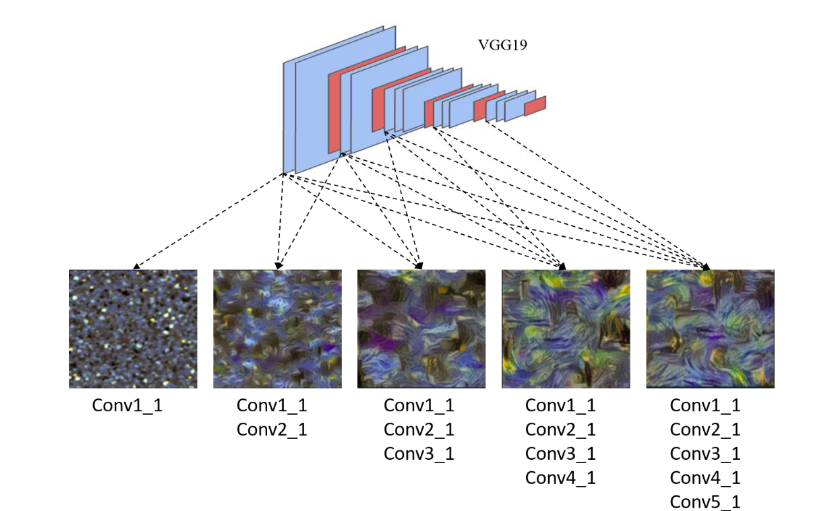


图3-4 风格迁移网络的结构

卷积神经网络中的每一层都会对图像产生不同的特征表示，在我们的研究中我们发现，为了获得风格表示的最佳结果，我们必须将几层的结果组合在一起，而不是只使用一层。对于内容表示，我们发现合并不同层的结果没有任何好处，在上图中，底部行显示内容信息即使在更高层中也保留了，尽管详细的像素信息丢失了。因此，在我们的算法中，我们将使用 ‘conv5\_2’ 层作为内容表示，使用 ‘conv1\_1’、‘conv2\_1’、‘conv3\_1’、‘conv4\_1’、‘conv5\_1’层作为样式表示。

**（1）首先是5层特征图的样式层：**

self.conv1style\_layers = ['block1\_conv1',  
 'block2\_conv1',  
 'block3\_conv1',  
 'block4\_conv1',  
 'block5\_conv1'  
 ]

**（2）然后是内容层：**

content\_layers = ['block5\_conv2']

**（3）将图像加载到 vgg19 网络中：**

def load\_and\_process\_img(path\_to\_img):  
 img = load\_img(path\_to\_img)  
 img = tf.keras.applications.vgg19.preprocess\_input(img)  
 return img

**（4）加载 VGG19 模型并访问中间层返回输出样式和内容中间层**

def get\_model():  
 import ssl  
ssl.\_create\_default\_https\_context = ssl.\_create\_unverified\_context  
vgg = tf.keras.applications.vgg19.VGG19(include\_top=False, weights='imagenet')  
vgg.trainable = False  
style\_outputs = [vgg.get\_layer(name).output for name in style\_layers]  
content\_outputs = [vgg.get\_layer(name).output for name in content\_layers]  
model\_outputs = style\_outputs + content\_outputs  
 return models.Model(vgg.input, model\_outputs)

**（5）之后是损失网络：**

要求成图片的内容要尽量和给定的内容图片类似，生成图片的风格要尽量与给定的风格图片类似。

**（6）计算内容损失**

def get\_content\_loss(base\_content, target):  
 return tf.reduce\_mean(tf.square(base\_content - target))

**（7）计算样式表示的Gram矩阵**

由于一张特征图可以表示某个卷积核对图片的理解，而多个特征图也只能代表多个分开的理解，不能代表总体的风格。我们需要一种方法把这些分开的理解整合成整体的理解。Gram矩阵大致是将任意两人的差异记录下来，于是将许多个体转为了一个整体：多个矩阵变成了一个矩阵。

def gram\_matrix(input\_tensor):  
 channels = int(input\_tensor.shape[-1])  
 a = tf.reshape(input\_tensor, [-1, channels])  
 n = tf.shape(a)[0]  
 gram = tf.matmul(a, a, transpose\_a=True)  
 return gram / tf.cast(n, tf.float32)

**（8）计算风格损失：**

def get\_style\_loss(base\_style, gram\_target):  
 height, width, channels = base\_style.get\_shape().as\_list()  
 gram\_style = gram\_matrix(base\_style)  
 return tf.reduce\_mean(tf.square(gram\_style - gram\_target))

# 计算总损失

def compute\_loss(model, loss\_weights, init\_image, gram\_style\_features, content\_features):  
 style\_weight, content\_weight = loss\_weights  
 model\_outputs = model(init\_image)  
 style\_output\_features = model\_outputs[:num\_style\_layers]  
 content\_output\_features = model\_outputs[num\_style\_layers:]  
 style\_score = 0  
 content\_score = 0  
 weight\_per\_style\_layer = 1.0 / float(num\_style\_layers)  
 for target\_style, comb\_style in zip(gram\_style\_features, style\_output\_features):  
 style\_score += weight\_per\_style\_layer \* get\_style\_loss(comb\_style[0], target\_style)  
 weight\_per\_content\_layer = 1.0 / float(num\_content\_layers)  
 for target\_content, comb\_content in zip(content\_features, content\_output\_features):  
 content\_score += weight\_per\_content\_layer \* get\_content\_loss(comb\_content[0], target\_content)  
 style\_score \*= style\_weight  
 content\_score \*= content\_weight  
 loss = style\_score + content\_score  
 return loss, style\_score, content\_score

# 使用Adam Optimizer

opt = tf.train.AdamOptimizer(learning\_rate=5, beta1=0.99, epsilon=1e-1)

**（9）迭代训练风格图片模型：**

当我们将内容和样式图像传递到VGG19网络中，获得它们的特征表示矩阵。从其特征表示中计算样式图像Gram矩阵后，开启一个迭代训练。

1. 从图像的特征表示中计算生成图像的Gram矩阵。

2. 计算内容损失:计算内容和生成的特征表示矩阵之间的均方误差。

3. 计算风格损失:计算风格和生成的Gram矩阵之间的均方误差。

4. 计算总损失:将内容损失和样式损失相加。

5. 使用优化算法计算梯度。

6. 从总损失中更新最佳损失和最佳图像。

# 四、实验结果分析

基于我们的研究，我们估计获得一个高质量图像需要算法运行1000次左右迭代，一个中等质量图像需要500次迭代，一个低质量的图像需要100次迭代。

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名 | 描述 | 预期结果 | 备注 |
| 不上传内容特征图 | 不上传内容特征图，点击生成艺术图 | 提示未上传内容特征图 | 确保用户上传内容特征图 |
| 不上传风格特征图 | 不上传风格特征图，点击生成艺术图 | 提示未上传风格特征图 | 确保用户上传风格特征图 |
| 不上传任何特征图 | 不上传任何特征图，点击生成艺术图 | 提示未上传内容特征图和风格特征图 | 确保用户上传内容特征图和风格特征图 |
| 上传内容特征图和风格特征图 | 上传内容特征图和风格特征图后，点击生成艺术图 | 正确运行，得到艺术图 |  |
| 选择输出质量 | 可选低中高。例如：选择输出低质量 | 得到一个低质量的艺术图 | 我们定义低质量为100次迭代 |
| 选择分辨率 | 例如选择512px的分辨率 | 得到分辨率为512x512的艺术图 |  |

### 4.1实验一

在这个实验中，我们选择了低质量水平，这意味着算法运行100次迭代，选择分辨率为512像素如图4-1所示，我们可以看到输出的图像质量不高，而且颜色效果不太好，如图4-2所示。



图4-1实验一图像风格迁移中



图4-2实验一图像风格迁移效果图

### 4.2实验二

在这个实验中，我们选择了中等质量水平，这意味着算法运行500次迭代，选择分辨率为512像素如图4-3所示，我们发现输出的图像质量比上次实验要好，但是颜色效果还是不满意，效果图如图4-4所示。



图4-3实验二图像风格迁移中



图4-4实验二图像风格迁移效果图

### 4.3实验三

在这个实验中，我们选择了高质量水平，这意味着算法运行1000次迭代，选择分辨率为1024像素获得更好的质量如图4-5所示，我们可以看到在这个实验中输出的图像是高色彩的质量，并与之前的实验结果相比较，这是最好的结果，效果图如图4-6所示。



图4-5实验三图像风格迁移中

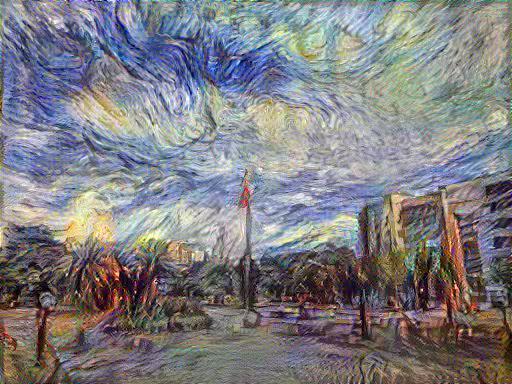


图4-6实验三图像风格迁移效果图

# 五、结论

如今图像风格迁移已经成为了一个热门的研究方向。基于神经网络的图像风格迁移在2015年由Gatys et al. 在两篇论文中提出：Gatys et al., 2015a和Gatys et al., 2015b。Gatys采用Gramian matrix计算那些不同局部特征的相关性，把它变成了一个统计模型，于是就有了一个不用手工建模就能生成纹理的方法。Gatys所做的改进是把两个不同领域的研究成果有机的结合了起来，做出了令人惊艳的结果。其实最让人惊讶的是纹理竟然能够和人们心目中认识到的图片的风格在很大程度上相吻合。（和真正的艺术风格有很大区别，但是看上去挺好看的。）从那之后对neural style的改进也层出不穷。基于模型迭代的方法是快速的风格迁移的方法，快速风格迁移只要数秒就可以迁移出一张图片，比原始风格迁移的速度要快很多。

# 六、参考文献

[1][Lample, et al., NeurIPS’19] Guillaume Lample, Alexis Conneau, Cross-lingual Language Model Pretraining, NeurIPS, 2019

[2] [Baziotis, et al., NAACL’19] Christos Baziotis, Ion Androutsopoulos, Ioannis Konstas, Alexandros Potamianos, “SEQ^3: Differentiable Sequence-to-Sequence-to-Sequence Autoencoder for Unsupervised Abstractive Sentence Compression”, NAACL 2019

[3] [Matt J. Kusner, et al., arXiv, 2016] Matt J. Kusner, José Miguel Hernández-Lobato, GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution, arXiv 2016

[4] <https://www.cnblogs.com/hichens/p/12355008.html>

[5] Gatys, L. A., Ecker, A. S., & Bethge, M. (2016). Image style transfer using convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2414-2423).

[6] Lempitsky, V. and C.V. Group, Instance Normalization: The Missing

Ingredient for Fast.

[7] Chuan Li, Michael Wand; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2479-2486

[8] [Jiaxian Guo, et al., AAAI, 2018] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, Jun Wang, Long Text Generation via Adversarial Training with Leaked Information, AAAI, 2018

[9] [Xu, et al., EMNLP, 2018] Jingjing Xu, Xuancheng Ren, Junyang Lin, Xu Sun, Diversity-Promoting GAN: A Cross-Entropy Based Generative Adversarial Network for Diversified Text Generation, EMNLP, 2018

[10] [Cyprien de Masson d'Autume, et al., arXiv 2019] Cyprien de Masson d'Autume, Mihaela Rosca, Jack Rae, Shakir Mohamed, Training language GANs from Scratch, arXiv 2019