离线神经网络任意风格迁移化的移动端实现

戴志豪，梁丹怡，彭 福

(南方科技大学 计算机科学与工程系,广东 深圳)

摘 要: 神经网络风格迁移指的是利用卷积神经网络提取的风格特征与内容特征，通过风格重建与内容重建的结合实现风格迁移的一类方法。目前市面上的风格迁移移动应用均是基于远程服务器运算的在线应用，且只能实现指定风格的迁移。本次实验将尝试在移动端 iOS 平台上首次实现离线的任意风格迁移，并提出了相应的系统结构设计与预期结果。

关键词: 神经网络风格迁移;风格重建;内容重建

The First Offline Arbitrary Neural Style Transfer Mobile Application on Android Platform

DAI Zhihao, LIANG Danyi, PENG Fu

(Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China)

Abstract: Neural Style Transfer is a class of methods that, through the combination of style reconstructions and content reconstructions, transfer one style from an image to another using high-level style features and content features extracted from convolutional neural networks. Currently, off-the-shelf mobile style transfer applications all rely on remote servers to perform the heavy computations and only provide a limited number of styles. In this experiment, we aims to implement the first offline arbitrary style transfer application on the Android mobile platform and lays out the corresponding system design and expected results.

Key words: neural style transfer; style reconstructions; content reconstructions

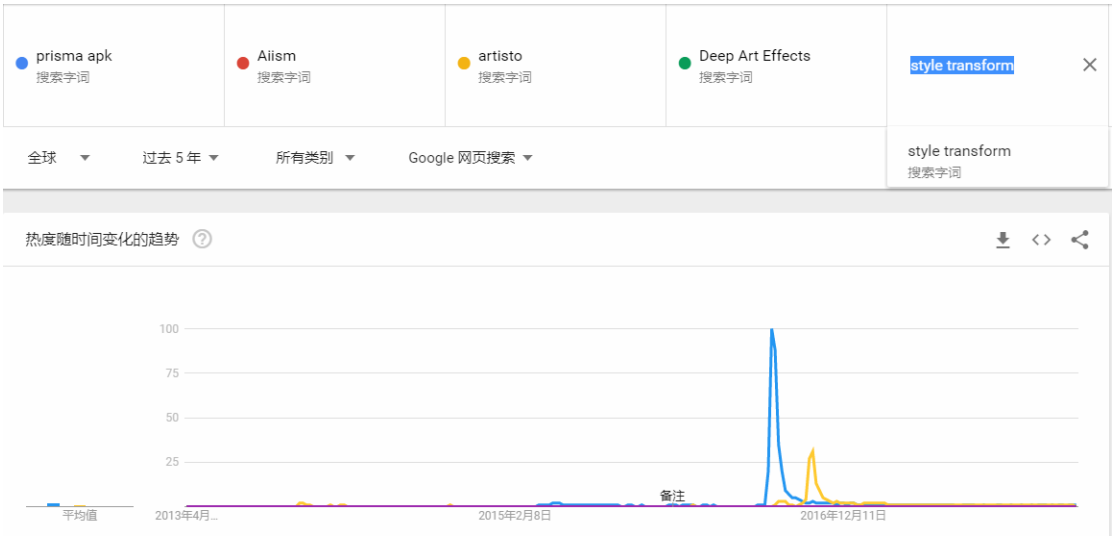
1 问题的定义

如今神经网络风格迁移化的移动端实现 Prisma 风靡全球，但是由于计算量大，现有的 App 实现都需要在线使用，即将图片上传到服务器，在服务器端的 DNN 完成风格迁移化后再传到手机端，从而在处理时间和必须联网两个方面限制了 App 的使用。风格迁移，又称纹理迁移，指的是将一张图片的风格转移至另一张图片或视频上的一类方法。一张图片，可以视作由内容和风格两部分组成。内容指的是图片中的物体、人物及其大小、位置、布局等空间属性，而风格指的是图片中除内容以外的视觉表现，包括纹理、颜色等属性。神经网络风格迁移指的是利用图片输入至神经网络的特征表示，实现将一张图片的风格转移至另一张图片，并保留后者的内容，生成新的图片的一类方法。我们将提供风格的图片称为“风格图片”，将提供内容的图片称为“内容图片”，并将生成的图片称为“生成图片”（或“结果图片”）。任意风格迁移指的是给定任意的风格图片和内容图片，不需要再额外训练模型，就能生产相应的目标图片。本课题拟实现一个能够离线使用的神经网络任意风格迁移化的 iOS 应用。

2 市场调研

2.1 市面上存在的产品

除了 prisma 之外，还有 artisto 、Aiism 美术滤镜、ArcSoft 虹软 、Deep Art Effects、深黑、vinci、GoArt 等产品。



图一：谷歌趋势 “prisma apk”,” Aiism”,” artisto”,” Deep Art Effects”“ style transform”词条热度比较

风格迁移的热度从 2016 年 6 月之后飙升，然后快速滑落。市面上目前有挺多风格迁移 app 的，但是目前最出名的 APP 还是 prisma。所知道的 APP 中，所有的 APP 都是提供内置的风格图片，不能实现任意风格迁移功能。目前市面上能找到的风格迁移产品，如表一所示。

产品名称	产品类型	风格数量	运行速度	图片是否需要裁剪	是否可以调整风格迁移程度	区域
Prisma	手机 APP	70	20 秒以上	否	是	俄罗斯
Vinci	手机 APP	20	2~3 秒	是	否	————
Artisto	手机 APP	30	2~3 秒	否	否	————
GoArt	手机 APP	70	5~8 秒	否	否	中国
Deep art effect	手机 APP	24	————	否	否	————
ArcSoft 中国	软件开发包	————	————	————	————	中国
Aiism 美术滤镜	网页端	————	————	————	————	中国
深黑	手机 APP	————	————	————	————	中国

表一 市面产品汇总

*上述产品难以调查用户量。其中 Aiism 美术滤镜网页不可用，深黑手机 app 已经停止开发维护
*使用场景：用户一般用于图片的美化。

2.2 Facebook 和Google在离线端风格迁移

随着风格迁移技术的发展和手机性能的提升，Facebook 和 Google 相继宣布在手机端上实现了离线风格迁移。在 Facebook 方面，Jia 等人利用 Caffe2Go，一个基于移动平台的深度学习框架，实现了离线的实时风格迁移，并在 iPhone 6s 上可以达到每秒 20 帧的视频输出。比较可惜的是，Facebook 并没有开源 Caffe2Go 的源代码，也使得我们无法对 Caffe2Go 的实时离线风格迁移实现进行分析。Facebook 声称 Caffe2Go 遵循了 Unix 的轻量化、模块化的开发哲学，使得框架的核心十分轻量，且可以方便地与各模块连接。在众多模块中，Caffe2Go 整合了 NNPack 库，利用了移动端手机的 CPU 的一项名为 NEON 的特性，大幅提升了 CPU 运算速度。在 iOS 设备上，Caffe2Go 还利用了 Metal 语言等特性进一步加速运算。

为了实现实时的风格迁移，Jia 等人[6]还对风格迁移模型进行了大量的优化，主要包括三个方面，减少卷

积层数、减少每一层的宽度以及调整图像的分辨率。由于卷积层的运算是风格迁移中运算量最大的部分，减少卷积层数和减少每一层的宽度将能够大幅减少卷积层的运算负担，从而提高运算速率。调整图像的分辨率由在迁移前降低输入图片的分辨率和在迁移后提高图片的分辨率两步组成，由于手机端相比 PC 端对分辨率的要求并不高，且分辨率的降低使得运算量相应降低，所以这一方法将能有效降低运算时间。

2.3 在Google方面，Vijaj等人在安卓平台上实现了离线多风格迁移的Demo程序，并公开了相应的源代码[11]。在具体的实现上，该程序采用Vincent等人[1]的多风格迁移方法，调用了TensorFlow的Java API中提供的TensorFlowInferenceInterface接口，将内容图片经过降低分辨率等处理后，直接输入至预先训练的模型中，直接得到了迁移后的结果。在模型的优化上，Google与Facebook的思路相近，采取减少卷积层数和减少每一层的宽度的方法，降低了模型的参数量和运算量，使得手机端的迁移运算成为可能。在Vincent等人的方法中，模型的参数量有40,000多个，而在Google的离线风格迁移程序中，参数量降到了不到1000个，大幅减轻了手机端的运算负担。

值得一提的是，上述两种离线风格迁移的实现都没有实现任意风格迁移。也就是说，在这一类风格迁移中，用户可选的风格的种类是有限的，而用户无法自定义风格图片。

2.4 需求和结论

经过调查，我们进行了需求分析，如表二所示。

	需求	证据
用户	1）提高软件的可用性 2）个人隐私安全 3）可自定义图片风格	1） 离线后无法使用 2） 将图片上传到服务器，具有隐私泄露隐患 3） 只能选择产品内置的风格，无法自定义风格
工业界	1）提高产品的质量 2）用户隐私安全 3）减少服务器的压力	1）照片处理时间无法确定 2）用户的隐私有泄露的风险 3）当 app 在线人数多的时候，服务器压力大
学术界	DNN 压缩和风格迁移化的结合	1） 神经网络离线嵌入式部署是当前热门的研究方向 2） DNN 压缩应用于风格迁移化，能够减少其计算量，有望在移动端实现离线风格迁移化

表二 需求分析

结论：经过市场调查和需求分析，我们认为离线的风格迁移移动端实现能够提高同类产品的质量，提高用户体验，具有一定的市场价值。

3 文献综述

90 年代中期-2015 年，人们开始尝试从计算机的角度探究用特定的艺术风格去重新绘制已有图像。这一阶段的图像风格迁移，主要的实现技术是非真实感渲染(NPR, Non-photorealistic Rendering)[7]：通过针对某种特定风格（油画、卡通等）的图像进行分析，建立该风格的统计学模型，再改变图像使之在统计特征上符合所

建立的风格数学模型。这样可以通过建立精细的数学模型得到更好的风格迁移效果，随着发展也逐渐成为计算机图形学的一个分支，但局限于特定的风格，应用场景有限。

随着卷积神经网络在图像分类识别的应用，2015 年 Gatys 等人[2] 提出了基于卷积神经网络的风格迁移化算法，风格迁移化得到进一步的概念化和体系化。此后的图像风格迁移研究工作均基于卷积神经网络进行，取得了较好的效果和更高的灵活性。算法大体而言分为两类[7]，一类基于图像优化[9][10]，如最大平均差异(MMD, Maximum mean discrepancy)和马尔科夫随机场(MRF, Markov random field)，通过迭代更新图像像素，运算过程较慢；一类基于预先设置的生成模型优化[5][8]，运算过程较快。在后者中，Johnson 等人[8]提出的方法加速了单风格迁移过程，使之可以达到实时效果；Huang 等人[5] 和 Ghiasi 等人[4]提出的方法实现了任意风格迁移效果。

3.1 风格重建和内容重建

基于深度卷积神经网络的风格迁移方法，受到深度卷积神经网络在 ImageNet 上的成功应用及其强大的可迁移性的启发，最早由 Gatys 等人提出[2][3]。Gatys 等人观察到[3]，在用于物体识别的卷积神经网络中，层级越高的特征映射（feature map），代表的物体信息便越高级（high-level），对图片的内容更加敏感，而对图片的外表越不敏感。相反，层级越低的特征映射，代表的物体信息便越低级（low-level），也就保留了更多图片的外表信息。通过将风格迁移分为风格重建和内容重建两部分，以白噪声图片为初始值，优化由特定子集的网络层激活值计算的风格损失和由高层级的激活值计算的内容损失，便可以实现风格迁移。

3.1.1 内容重建

为了可视化图片的信息，这些信息被编码在不同的层中，我们在一个白噪声图片上使用梯度下降法去找到另外一张图片，这张图片跟原始图片的特征反馈吻合。所以让$\vec{p}$和$\vec{x}$成为原始图片和被生成的图片并且 P^l 和 F^l 分别在 l 层的特征表达。Gatys 等人[3]在两个特征表达中间定义了针对内容的平方误差损失函数:

$$L_{\text{content}}(\vec{p},\vec{x},l)=\frac{1}{2}\sum\left(F_{ij}^{l}-P_{ij}^{l}\right)^{2} \tag{1}$$

这个损失函数相对于在 l 层的激活的导数为：

$$\frac{\partial \mathcal{L}_{\text{content}}}{\partial F_{ij}^{l}}=\begin{cases}\left(F^{l}-P^{l}\right)_{ij} & if\ F_{ij}^{l}>0\\ 0 & if\ F_{ij}^{l}<0.\end{cases} \tag{2}$$

从公式（2）可以得到图像$\vec{x}$的梯度能够根据标准差反向传播来计算。因此，Gatys 等人能够改变初始的随机图片$\vec{x}$直到它在特定的 CNN 层生成像原始图片$\vec{p}$一样相同的反馈。

3.1.2 风格重建

在神经网络的每一层 CNN 反馈的顶部，Gatys 等人[3]建立了一个计算不同过滤器反馈之间关系的一个风格表达。这些特征关系是 gram 矩阵$G^l \in \mathcal{R}^{N_l \times N_l}$给出的，$G_{ij}^{l}$是在 l 层向量化特征映射 i 和 j 的内积：

$$G_{ij}^{l}=\sum F_{ik}^{l}F_{jk}^{l}. \tag{3}$$

定义 $\vec{a}$ 和 $\vec{x}$ 分别为原图片和将要生产的图片，则 A^l 和 G^l 分别为第 l 个卷积层对应的风格特征表示矩阵。第 l 个卷积层的风格特征损失为：

$$E_l=\frac{1}{4N_l^2M_l^2}\sum\left(G_{ij}^{l}-A_{ij}^{l}\right)^{2} \tag{4}$$

定义 w_l 为第ℓ个卷积层的风格损失的权重（weighting factor），则总的风格损失为：

$$\mathcal{L}_{\text{style}}(\vec{a},\vec{x})=\sum_{l=0}^{L}w_lE_l \tag{5}$$

结合内容重建与风格重建，令$\vec{p}$为内容图片，$\vec{a}$风格图片，$\vec{x}$为生成的图片，定义总的损失函数为:

$$\mathcal{L}_{\text{total}}(\vec{p},\vec{a},\vec{x}) = \alpha\mathcal{L}_{\text{content}}(\vec{p},\vec{x}) + \beta\mathcal{L}_{style}(\vec{a},\vec{x}) \quad (6)$$

其中，α和β分别为内容重建与风格重建的权重。

3.2 快速风格迁移

为了解决 Gatys 等人的优化方法过慢的缺陷，Johnson 等人提出前向卷积神经网络生成目标图片的快速风格迁移方法[8]。在 Johnson 的方法中，风格迁移系统分为两部分，一部分是图片转换网络（Image Transform Network），另一部分是损失网络（Loss Network）。

损失网络与 Gatys 等人的方法相似，采用了在 ImageNet 上预训练的 VGG-16 网络，用于计算损失函数的定义为：

$$\mathcal{L}_{\text{total}}(\vec{p},\vec{a},\vec{x}) = \alpha\mathcal{L}_{content}(\vec{p},\vec{x}) + \beta\mathcal{L}_{style}(\vec{a},\vec{x}) + l_{TV}\mathcal{L}_{TV}(\vec{x}) \quad (7)$$

其中 $\mathcal{L}_{\text{TV}}$（$\vec{x}$）是总变分正则化项（total variation regularizer），表示图片的空间不平滑度。

图片变化网络是一个残差（Residual）卷积神经网络，用于将输入图片变换为目标图片。他们定义 $\vec{x}$ 为输入图片，$\vec{y}$为通过$\vec{y} = f_w(\vec{x})$变换得到的输出图片。W 是神经网络中的权重。在训练时，损失网络的参数固定不变，图片变换网络用梯度下降法最小化损失函数。

$$\mathrm{W}^* = \arg\min_{w} E_{\vec{x}}\left[\mathcal{L}_{total}(\vec{p},\vec{a},\vec{x})\right] \quad (8)$$

由于在风格迁移时，Johnson 等人的方法只需要一次前馈就可以得到目标图片，相比 Gatys 等人的通过优化损失函数实现的方法，能达到 1000 倍以上的加速。

3.3 任意风格迁移

3.3.1 Huang 等人的方法

Huang 等人提出的 AdaIN-style 和 Zhang 等人提出的 MSG-Net 都很好地解决了这一问题[5]。由于 AdaIN-style 的计算量更小，模型更为轻便，更适合在移动端部署，这里我们只对 AdaIN-style 进行介绍。AdaIN-style 的风格迁移网络主要由三部分组成，VGG 编码器、AdaIN 模块和解码器。

VGG 编码器是将风格图片和内容图片输入至预训练的 VGG 卷积神经网络中，得到不同层级的特征映射。我们用 $x \in R^{N\times C\times H\times W}$表示输入（通常是某一层的特征映射），$N$ 表示样本个数，C 表示频道数目，W 和 W 表示输入层的宽和高。对于每一个频道和样本，我们都计算其平均值 $\mu_{nc}(x)$ 和标准差 $\sigma_{nc}(x)$。

$$\mu_{nc}(x) = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{nchw} \quad (9)$$

$$\sigma_{nc}(x) = \sqrt{\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{nchw} - \mu_{nc}(x))^2 + \epsilon} \quad (10)$$

AdaIN 模块实现的功能较为简单，其接收内容输入的特征映射 x 和 风格输入的特征映射 y，并通过仿射变换将 x 的平均值与方差与 y 对齐，得到目标图片的特征映射。

$$t = AdaIN(x,y) = \sigma(y)\left(\frac{x-\mu(x)}{\sigma(x)}\right) + \sigma(y) \quad (11)$$

解码器模块是一个卷积神经网络，负责将目标图片的特征映射解码为目标图片 $T(c,s)$。

$$T(c,s) = g(t) \quad (12)$$

为了训练解码器，我们定义了损失函数 L。

$$L = L_c + \lambda L_s \tag{13}$$

其中，内容损失 L_c 的定义是目标图片的特征映射（即 AdaIN 模块的输出）与生成图片间的欧几里得距离。

$$L_c = \left\| f(g(t)) - t \right\|_2 \tag{14}$$

风格损失 L_s 定义为目标图片的特征映射的平均值与方差和生成图片间的欧几里得距离，其中 ϕ_i 表示 VGG-19 网络中的网络层。

$$L_s = \sum_{i=1}^{L} \|\mu(\phi_i(g(t))) - \mu(\phi_i(s))\| \sum_{i=1}^{L} \|\sigma(g(t))) - \sigma(\phi_i(s))\| \tag{15}$$

通过优化损失函数 L ，我们便得到了一个针对目标图片的特征映射的解码器。

3.3.2 Ghiasi 等人的方法

Ghiasi 等人[4]提出的模型结构，在风格迁移网络 T 中嵌入了一个风格预测网络 P 以处理风格图片。风格预测网络对输入的风格图片进行启发式学习，得到一个可从特征空间上代表风格的向量 $\vec{S}$ ；并将向量输出到风格迁移网络中参与激活，以生成最终的图片。

3.3.2.1 损失函数

定义 c 为输入的内容图片，s 为输入的风格图片，x 为输出的结果合成图片，则训练风格迁移网络 T 时定义的优化目标为：

$$\min_x L_c\,(x,c) \;+\; \lambda_s\, L_s\,(x,s) \tag{16}$$

此处$L_c\,(x,c)$为内容损失函数，$L_s\,(x,s)$为风格损失函数，λ_s是与风格损失相关的拉格朗日乘子。计算损失函数时采用了 VGG 网络，其中一部分低层 S 提取的特征来代表图片的风格，一部分高层 C 提取的特征来代表图片的内容。

风格损失函数定义如下：

$$\mathrm{L}_s\,(x,s) \;=\; \sum_{i\in S} \frac{1}{n_i} \left|\left| G\,[\,f_i\,(x)] - G\,[\,f_i\,(s)]\,\right|\right|_F^2 \tag{17}$$

对于低层集合 S 中的每一层 i，$f_i\,(x)$为该层的网络激活值；$G\,[\,f_i\,(x)]$为该层的格拉姆矩阵，用特征之间的关系代表图像的风格信息。风格损失函数计算了输出的图片与风格图片在底层特征关系上的平均距离，作为输出与风格图片的差异。

内容损失函数定义如下：

$$\mathrm{L}_c\,(x,c) \;=\; \sum_{j\in C} \frac{1}{n_j} \left|\left| f_j\,(x) - f_j\,(c)\,\right|\right|_2^2 \tag{18}$$

对于高层集合 C 中的每一层 j，$f_j\,(x)$为该层的网络激活值，用高层特征代表图像的内容。内容损失函数计算了输出的图片与内容图片在高层特征上的平均距离，作为输出与内容图片的差异。

3.3.2.2 实例归一化

风格预测网络 P 从风格图片 s 中学习均值与标准差信息，最终输出一个包含各通道均值参数β_s和标准差参数γ_s的高维向量$\vec{S}$。对于风格生成网络 T 中的每一个通道 z，定义 μ 和 σ 分别为均值和标准差，对其作以下

线性变换：

$$\tilde{z} = \gamma_s \left(\frac{z-\mu}{\sigma} \right) + \beta_s \qquad (19)$$

其中β_s和γ_s为$\vec{S}$在该通道的均值和标准差参数。

目前，谷歌、Facebook 等都已实现了在移动端离线运行的单风格迁移，但并未能实现任意风格迁移。在本次项目中，我们希望将 Ghiasi 等人的网络移植至移动平台（iOS 系统）上，使得更多人都能轻松地使用图片的任意风格迁移，运用艺术品、画作、影片截图等作为风格图片，对生活中拍摄、记录的图片进行更加多样化的二次创造。

4 课题实施计划

4.1 知识准备

通过文献检索了解风格迁移化领域的发展历史和现状，挑选其中具有代表性且与本课题预期目标相关性较大的文献进行学习，使得在 PC 端可以简单复现任意风格迁移化功能相关文献的算法。

4.2 需求分析

参考现有的离线深度学习在离线部署和移动端部署的应用，根据其在网络结构、模型参数等方面的条件，结合预期结果预计最终模型的大小，内存、运行速率以及代码量。

4.3 软件开发

根据系统结构设计，在选定的平台（iOS）上进行应用的初步开发，选择小型的深度学习网络框架（TensorFlow）实现文献的算法，并在服务器进行模型训练。

4.4 调试

模拟运行或真机调试，并根据运行情况进行网络压缩、图片压缩和代码优化，使得应用达到实时的效果，同时兼顾更精细的风格迁移化效果。

5 移动端实现

我们参照 Ghiasi 等人提出的实时任意风格迁移模型[3]，使用谷歌的开源神经网络框架 Tensorflow 在桌面端训练并得到可在移动端上运行的模型文件。我们在 iOS 上实现离线的神经网络风格迁移应用。该移动应用将利用谷歌的面向移动平台的谷歌 Tensorflow API，使用在桌面端预训练的风格迁移 CNN 模型文件，将用户提供的风格图片与内容图片作为输入，得到风格迁移后的图片。

在完成模型在 PC 端上的训练后，我们成功将模型部署到 iOS 平台上。由于在 PC 端上我们使用 TensorFlow 定义并训练模型，而 TensorFlow 在 iOS 平台上的支持尚不完善，且依赖于 TensorFlow 底层的 C++接口，较为繁琐，因此我们选择了将 TensorFlow 模型转换为 CoreML 模型后，再在移动端上运行。由于 CoreML 是苹果 iOS 原生的机器学习框架，并加入到 iOS 开发工具包中，因此极大地简化了在 iOS 上的开发过程。我们的具体实现过程如下：

1. 将 TensorFlow 模型定义文件导出。

由于 TensorFlow Slim 模块在训练模型时，默认只会保存训练后的每个节点间的权重，而不保存每个节点的连接关系和输入输出定义（即网络的静态图）。由于 CoreML 要求模型是静态图，因此我们需要将动态定义的模型文件先转换为静态图模型文件。静态图模型文件大小为 44Mb。

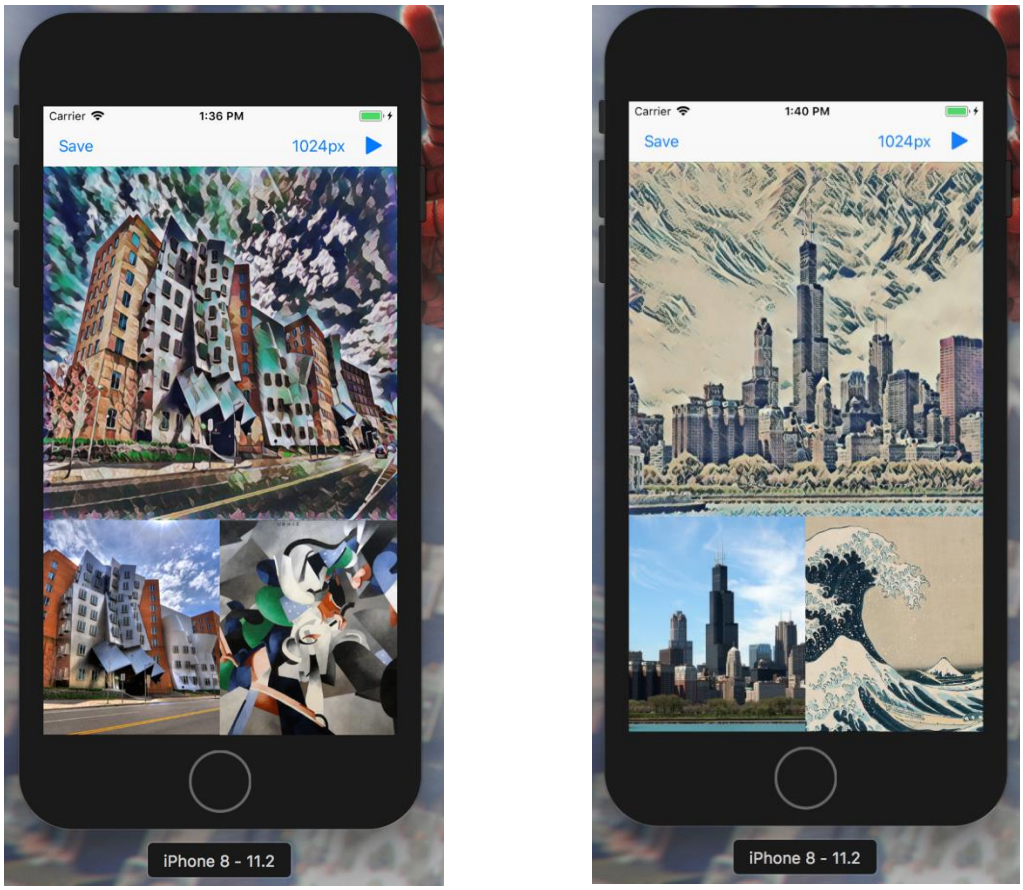
2. 将 TensorFlow 模型转换为 CoreML 模型文件。

我们使用 TF-CoreML（https://github.com/tf-coreml/tf-coreml）工具，将 TensorFlow 的静态图模型转换 CoreML 可以运行的模型文件。我们固定 P 网络的输入为 256x256x3 的风格图片，针对 T 网络的不同输入，

分别定义输入为 1024x1024x3, 512x512x3, 256x256x3 的内容图片的 T 网络，并导出为不同的模型文件，以便对比不同分辨率下模型在移动端上的表现。

3. iOS 上实现任意风格迁移

利用导出的模型文件，我们开发了一个 iOS 应用（SUStylize），它可以输入任意的内容图片和风格图片，并根据用户的设定，输出分辨率为 1024x1024x3, 512x512x3, 256x256x3 的迁移结果。



图二 任意风格的手机端 App SUStylize

6 预期结果

6.1 用户使用效果

用户可以使用本应用实现以下功能和步骤：

1） 从相册选择一张内容图片
2） 将图片进行裁剪
3） 从相册选择一张风格图片
4） 在 10 秒内完成任意风格迁移化并输出显示
5） 用户可以选择保存结果图片到相册

内容图片	风格图片	结果图片

表三 应用预期效果图

6.2 应用运行效果

	Prisma	本应用预期
运行内存占用	64-256MB	最高可达 700MB-800MB
图片处理时间	5-30 秒，有时网络连接超时	10 秒内
输出图片大小	小于 100KB	压缩图片分辨率至大小在 100KB 以内

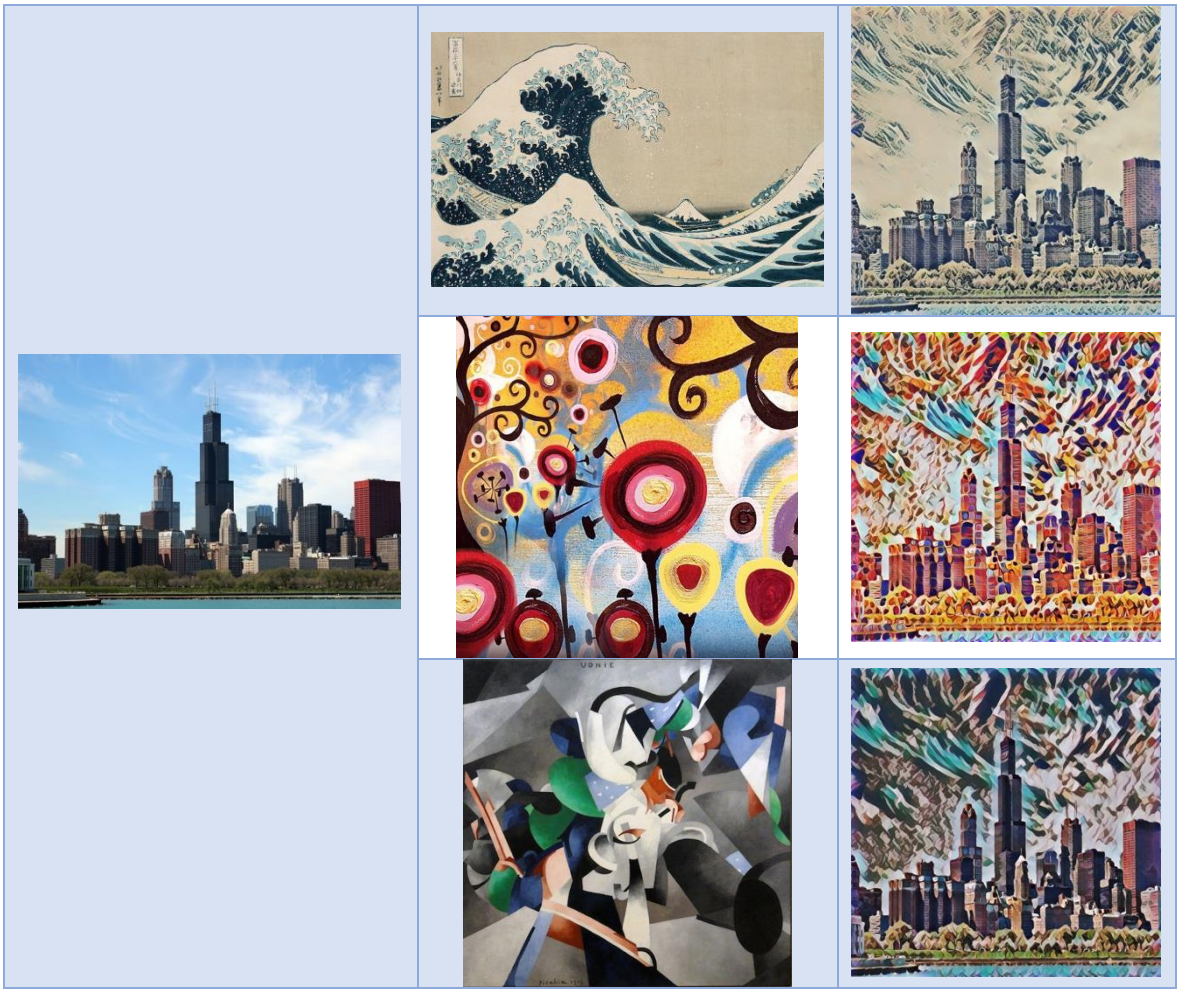
表四 与同类联网应用 Prisma 对比

7 最终结果

7.1 不同风格图片的迁移结果

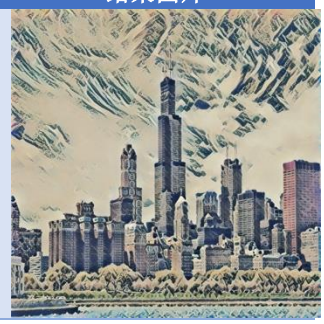
我们固定结果图片的输出为 1024x1024x3，固定内容图片为芝加哥城市，对比了 SUStylize 在不同风格图片下的迁移效果。

内容图片	风格图片	结果图片



7.2 不同内容图片的迁移结果

我们固定结果图片的输出为 1024x1024x3，固定风格图片为神奈川冲浪里，对比了 SUStylize 在不同内容图片下的迁移效果。

风格图片	内容图片	结果图片
		

可以看到，不管是固定风格还是固定内容，我们的离线任意风格迁移 App SUStylize 都取得了非常好的迁移结果，也实现了预期的目标。值得一提的是，它也是全球第一款实现离线任意风格迁移的移动端应用。

References:

[1] Dumoulin V, Shlens J, Kudlur M. A learned representation for artistic style[J]. CoRR, abs/1610.07629, 2016, 2(4): 5.

[2] Gatys L A, Ecker A S, Bethge M. A neural algorithm of artistic style[J]. arXiv preprint arXiv:1508.06576, 2015.

[3] Gatys L, Ecker A S, Bethge M. Texture synthesis using convolutional neural networks[C]//Advances in Neural Information Processing Systems. 2015: 262-270.

[4] Ghiasi G, Lee H, Kudlur M, et al. Exploring the structure of a real-time, arbitrary neural artistic stylization network[J]. arXiv preprint arXiv:1705.06830, 2017.

[5] Huang X, Belongie S. Arbitrary style transfer in real-time with adaptive instance normalization[J]. CoRR, abs/1703.06868, 2017.

[6] Jia Y, Vajda P. Delivering real-time AI in the palm of your hand[EB/OL]. https://code.facebook.com/posts/196146247499076/delivering-real-time-ai-in-the-palm-of-your-hand/, 2016-11-08/2018/04/05.

[7] Jing Y, Yang Y, Feng Z, et al. Neural style transfer: A review[J]. arXiv preprint arXiv:1705.04058, 2017.

[8] Johnson J, Alahi A, Fei-Fei L. Perceptual losses for real-time style transfer and super-resolution[C]//European Conference on Computer Vision. Springer, Cham, 2016: 694-711.

[9] Li C, Wand M. Combining markov random fields and convolutional neural networks for image synthesis[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 2479-2486.

[10] Y. Li, N. Wang, J. Liu, and X. Hou. Demystifying neural style transfer. ArXiv e-prints, Jan. 2017.

[11] https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/android