MakeGoodFood

Marvin Yang & Kehua Lei

https://github.com/poopoomarvin/MakeGoodFood

Part 1: Generate text using a neural network

Use GPT-2 Model to generate food reviews

1. Find an online dataset of food reviews.

   https://www.kaggle.com/snap/amazon-fine-food-reviews

2. Good food, good reviews.

   Extract reviews with 4&5 stars.

Use GPT-2 Model to generate food reviews

3. Config

learning rate:1e-4

steps:1000

4. Generate text

temperature:0.7

Use GPT-2 Model to generate food reviews

Examples:

input prefix: The sandwich

The sandwich is very tasty. The only reason I gave it 4 stars is because of the price.

Part 2: Generate multimedia output using a deep learning neural network

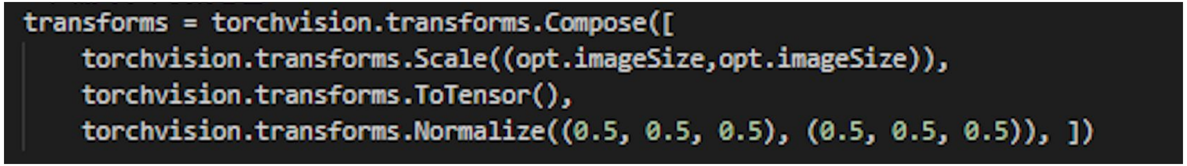
Section 1: Train a DCGAN model to generate food images

1. Find an online dataset of food images.

   https://github.com/karansikka1/iFood_2019

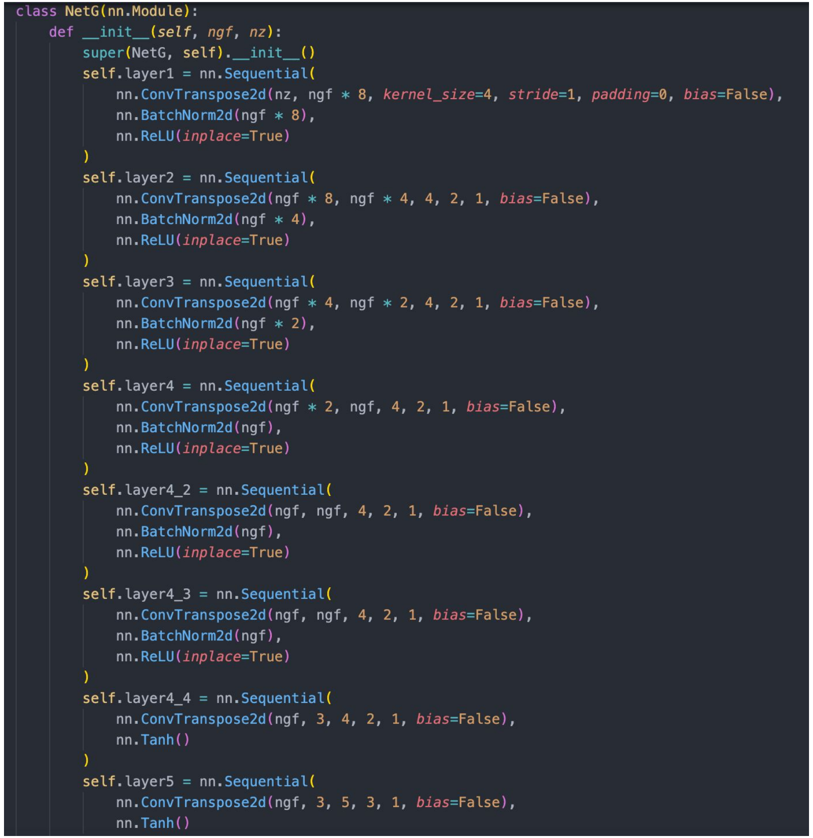
   118,475 training images

2. Cut images into the same size (96*96)

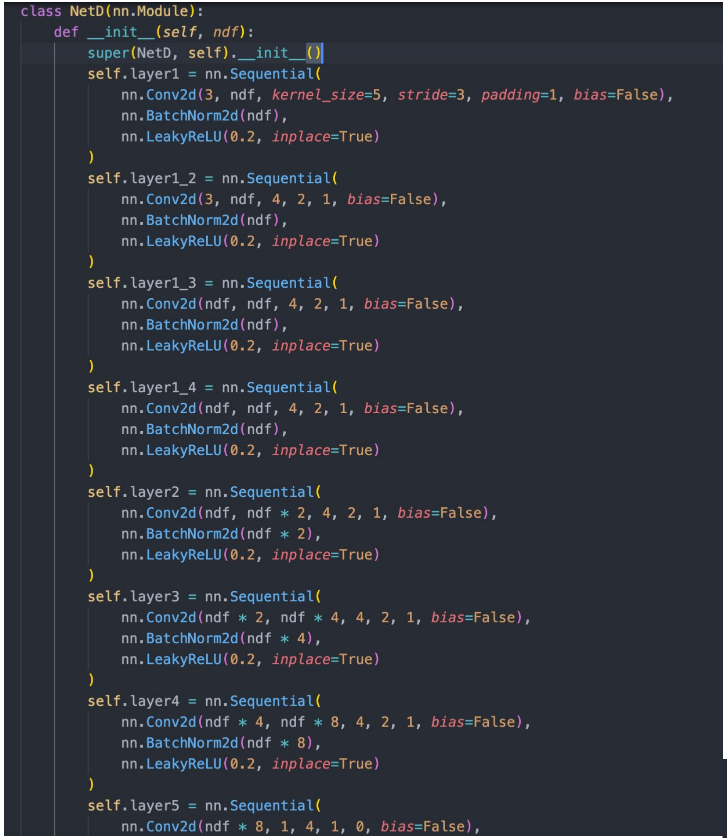


```
transforms = torchvision.transforms.Compose([
    torchvision.transforms.Scale((opt.imageSize,opt.imageSize)),
    torchvision.transforms.ToTensor(),
    torchvision.transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)), ])
```

3. Train a DCGAN

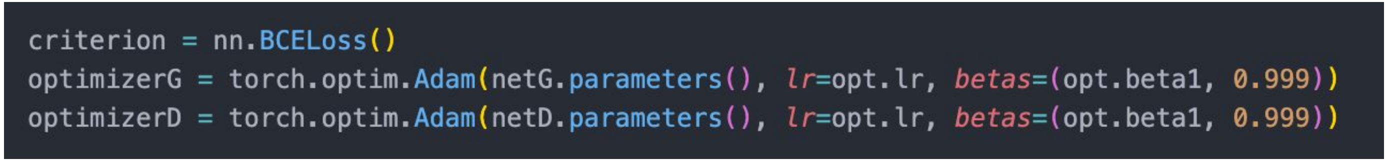


```python
class NetG(nn.Module):
    def __init__(self, ngf, nz):
        super(NetG, self).__init__()
        self.layer1 = nn.Sequential(
            nn.ConvTranspose2d(nz, ngf * 8, kernel_size=4, stride=1, padding=0, bias=False),
            nn.BatchNorm2d(ngf * 8),
            nn.ReLU(inplace=True)
        )
        self.layer2 = nn.Sequential(
            nn.ConvTranspose2d(ngf * 8, ngf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 4),
            nn.ReLU(inplace=True)
        )
        self.layer3 = nn.Sequential(
            nn.ConvTranspose2d(ngf * 4, ngf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 2),
            nn.ReLU(inplace=True)
        )
        self.layer4 = nn.Sequential(
            nn.ConvTranspose2d(ngf * 2, ngf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf),
            nn.ReLU(inplace=True)
        )
        self.layer4_2 = nn.Sequential(
            nn.ConvTranspose2d(ngf, ngf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf),
            nn.ReLU(inplace=True)
        )
        self.layer4_3 = nn.Sequential(
            nn.ConvTranspose2d(ngf, ngf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf),
            nn.ReLU(inplace=True)
        )
        self.layer4_4 = nn.Sequential(
            nn.ConvTranspose2d(ngf, 3, 4, 2, 1, bias=False),
            nn.Tanh()
        )
        self.layer5 = nn.Sequential(
            nn.ConvTranspose2d(ngf, 3, 5, 3, 1, bias=False),
            nn.Tanh()
```

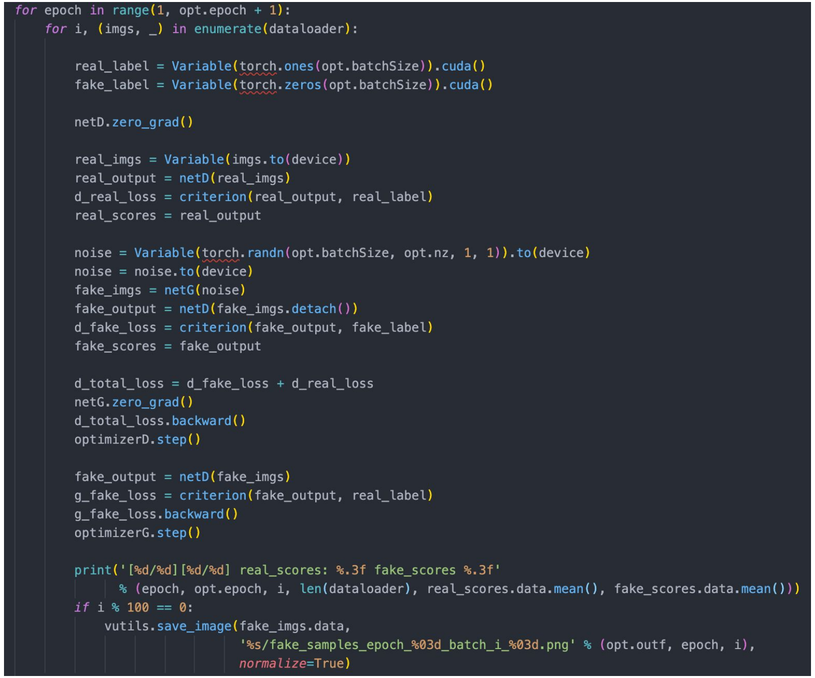


```python
class NetD(nn.Module):
    def __init__(self, ndf):
        super(NetD, self).__init__()
        self.layer1 = nn.Sequential(
            nn.Conv2d(3, ndf, kernel_size=5, stride=3, padding=1, bias=False),
            nn.BatchNorm2d(ndf),
            nn.LeakyReLU(0.2, inplace=True)
        )
        self.layer1_2 = nn.Sequential(
            nn.Conv2d(3, ndf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf),
            nn.LeakyReLU(0.2, inplace=True)
        )
        self.layer1_3 = nn.Sequential(
            nn.Conv2d(ndf, ndf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf),
            nn.LeakyReLU(0.2, inplace=True)
        )
        self.layer1_4 = nn.Sequential(
            nn.Conv2d(ndf, ndf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf),
            nn.LeakyReLU(0.2, inplace=True)
        )
        self.layer2 = nn.Sequential(
            nn.Conv2d(ndf, ndf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 2),
            nn.LeakyReLU(0.2, inplace=True)
        )
        self.layer3 = nn.Sequential(
            nn.Conv2d(ndf * 2, ndf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 4),
            nn.LeakyReLU(0.2, inplace=True)
        )
        self.layer4 = nn.Sequential(
            nn.Conv2d(ndf * 4, ndf * 8, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 8),
            nn.LeakyReLU(0.2, inplace=True)
        )
        self.layer5 = nn.Sequential(
            nn.Conv2d(ndf * 8, 1, 4, 1, 0, bias=False),
```

3. Train a DCGAN

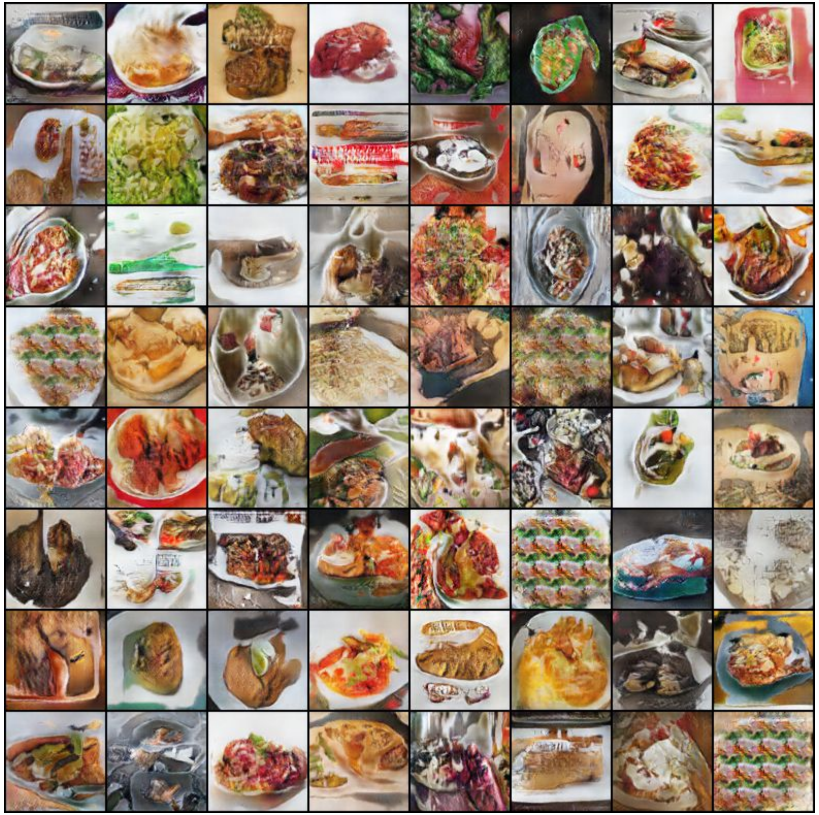


```
criterion = nn.BCELoss()
optimizerG = torch.optim.Adam(netG.parameters(), lr=opt.lr, betas=(opt.beta1, 0.999))
optimizerD = torch.optim.Adam(netD.parameters(), lr=opt.lr, betas=(opt.beta1, 0.999))
```



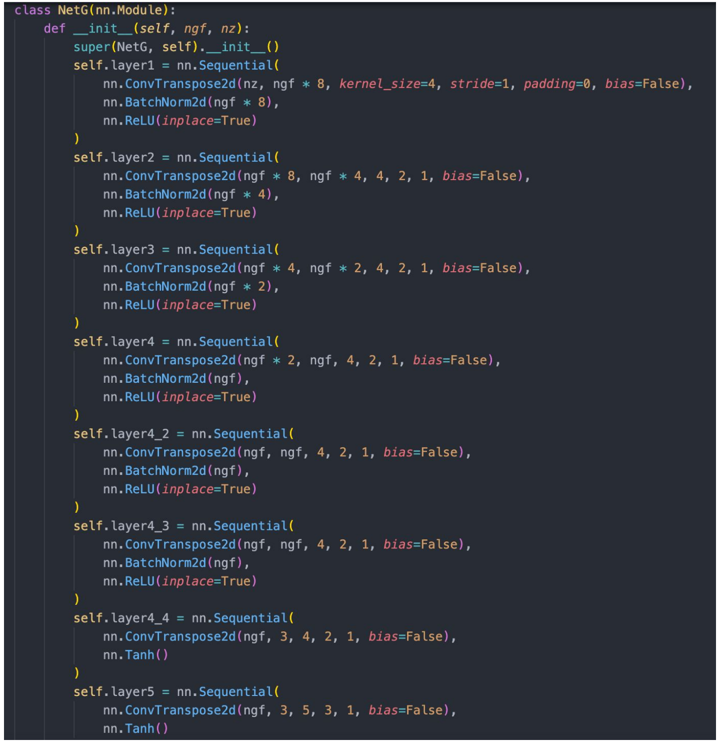
```
for epoch in range(1, opt.epoch + 1):
    for i, (imgs, _) in enumerate(dataloader):

        real_label = Variable(torch.ones(opt.batchSize)).cuda()
        fake_label = Variable(torch.zeros(opt.batchSize)).cuda()

        netD.zero_grad()

        real_imgs = Variable(imgs.to(device))
        real_output = netD(real_imgs)
        d_real_loss = criterion(real_output, real_label)
        real_scores = real_output

        noise = Variable(torch.randn(opt.batchSize, opt.nz, 1, 1)).to(device)
        noise = noise.to(device)
        fake_imgs = netG(noise)
        fake_output = netD(fake_imgs.detach())
        d_fake_loss = criterion(fake_output, fake_label)
        fake_scores = fake_output

        d_total_loss = d_fake_loss + d_real_loss
        netG.zero_grad()
        d_total_loss.backward()
        optimizerD.step()

        fake_output = netD(fake_imgs)
        g_fake_loss = criterion(fake_output, real_label)
        g_fake_loss.backward()
        optimizerG.step()

        print('[%d/%d][%d/%d] real_scores: %.3f fake_scores %.3f'
              % (epoch, opt.epoch, i, len(dataloader), real_scores.data.mean(), fake_scores.data.mean()))
        if i % 100 == 0:
            vutils.save_image(fake_imgs.data,
                              '%s/fake_samples_epoch_%03d_batch_i_%03d.png' % (opt.outf, epoch, i),
                              normalize=True)
```

4. Results



Section 1: Train a DCGAN model to generate food images

3. Train a DCGAN



```python
class NetG(nn.Module):
    def __init__(self, ngf, nz):
        super(NetG, self).__init__()
        self.layer1 = nn.Sequential(
            nn.ConvTranspose2d(nz, ngf * 8, kernel_size=4, stride=1, padding=0, bias=False),
            nn.BatchNorm2d(ngf * 8),
            nn.ReLU(inplace=True)
        )
        self.layer2 = nn.Sequential(
            nn.ConvTranspose2d(ngf * 8, ngf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 4),
            nn.ReLU(inplace=True)
        )
        self.layer3 = nn.Sequential(
            nn.ConvTranspose2d(ngf * 4, ngf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 2),
            nn.ReLU(inplace=True)
        )
        self.layer4 = nn.Sequential(
            nn.ConvTranspose2d(ngf * 2, ngf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf),
            nn.ReLU(inplace=True)
        )
        self.layer4_2 = nn.Sequential(
            nn.ConvTranspose2d(ngf, ngf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf),
            nn.ReLU(inplace=True)
        )
        self.layer4_3 = nn.Sequential(
            nn.ConvTranspose2d(ngf, ngf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf),
            nn.ReLU(inplace=True)
        )
        self.layer4_4 = nn.Sequential(
            nn.ConvTranspose2d(ngf, 3, 4, 2, 1, bias=False),
            nn.Tanh()
        )
        self.layer5 = nn.Sequential(
            nn.ConvTranspose2d(ngf, 3, 5, 3, 1, bias=False),
            nn.Tanh()
```

Section 2: Style Transfer Food Images

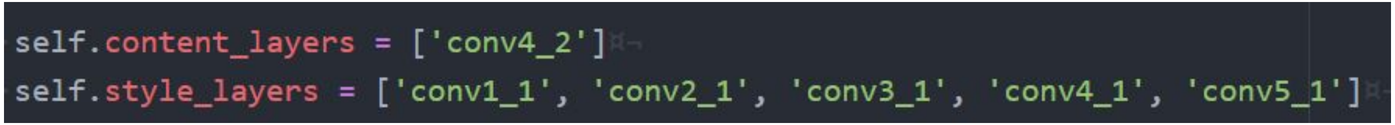
1. Find an online dataset of food images.

https://www.kaggle.com/vermaavi/food11

16643 food images grouped in 11 major food categories

Section 2: Style Transfer Food Images

2. VGG19 Model



```
self.content_layers = ['conv4_2']
self.style_layers = ['conv1_1', 'conv2_1', 'conv3_1', 'conv4_1', 'conv5_1']
```

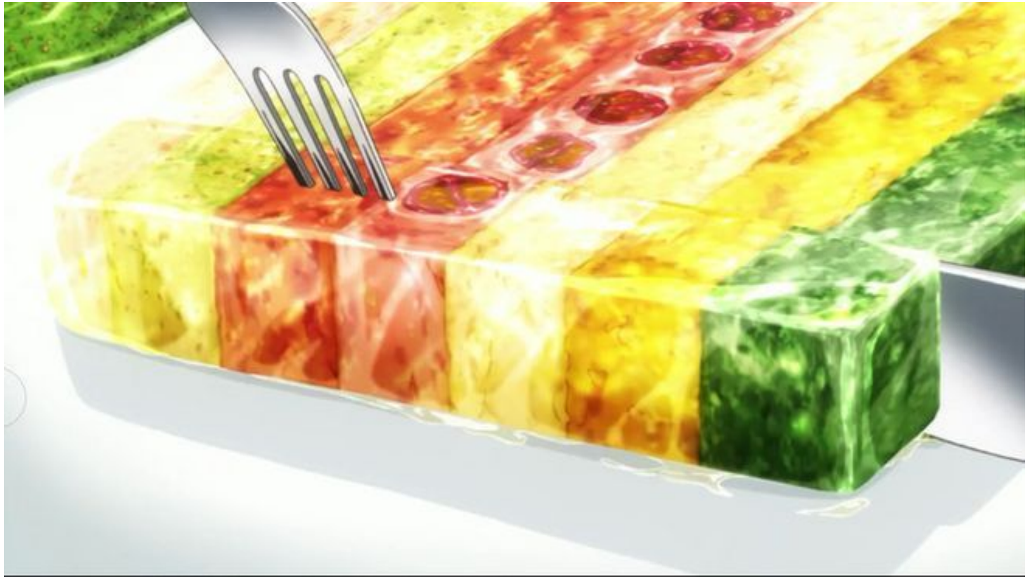
3. Config

Style weight:0.95 Content Weight:0.05

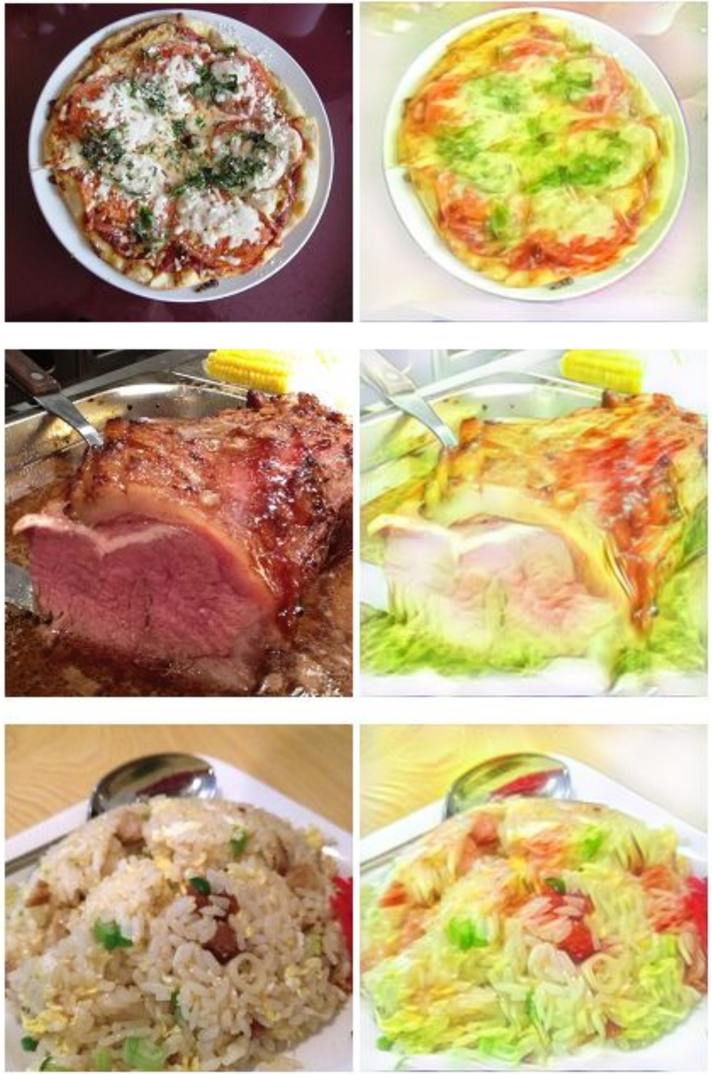
Learning rate:1e-3 Iterations:160k

Section 2: Style Transfer Food Images

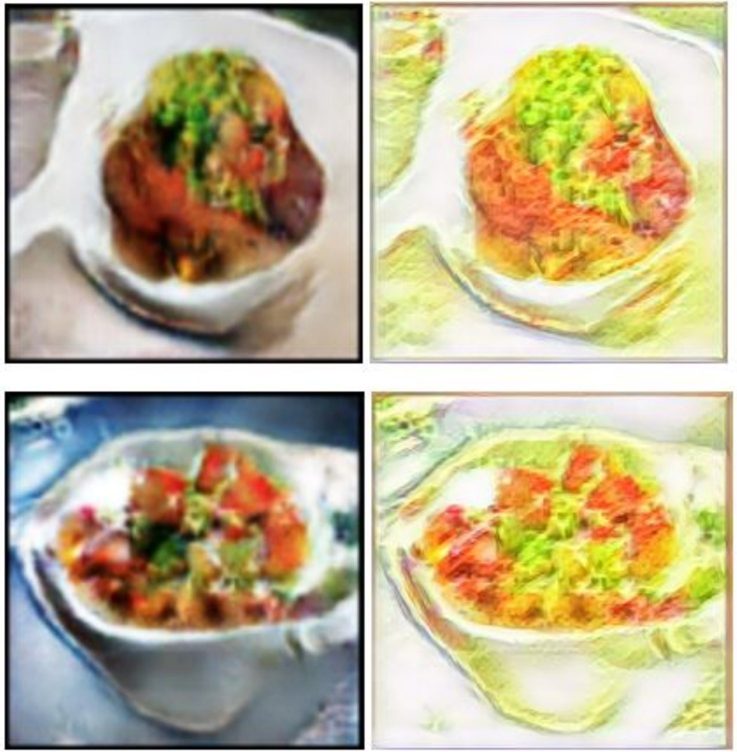
4. Style Image

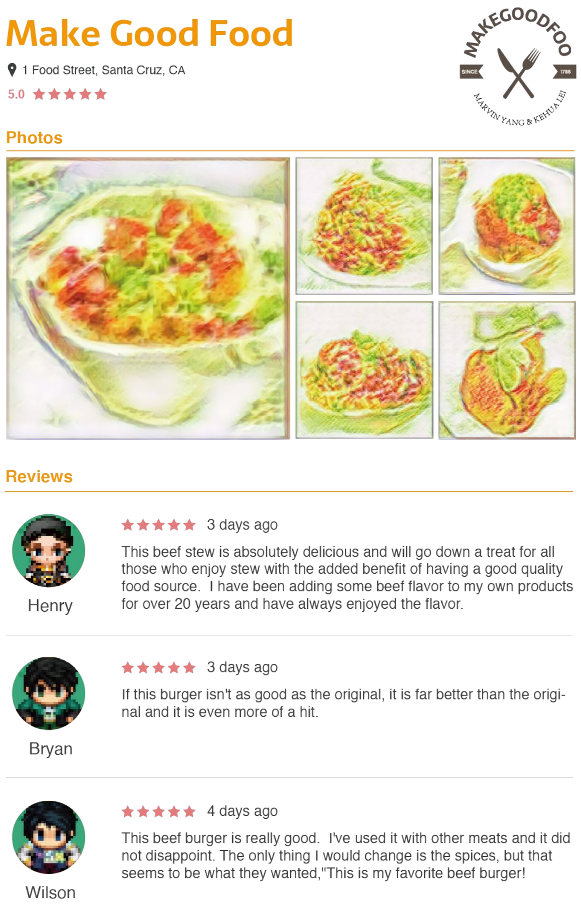


5. Results



5. Results





Make Good Food

1 Food Street, Santa Cruz, CA

5.0 ★★★★★

Photos

Reviews

Henry

★★★★★ 3 days ago

This beef stew is absolutely delicious and will go down a treat for all those who enjoy stew with the added benefit of having a good quality food source. I have been adding some beef flavor to my own products for over 20 years and have always enjoyed the flavor.

Bryan

★★★★★ 3 days ago

If this burger isn't as good as the original, it is far better than the original and it is even more of a hit.

Wilson

★★★★★ 4 days ago

This beef burger is really good. I've used it with other meats and it did not disappoint. The only thing I would change is the spices, but that seems to be what they wanted,"This is my favorite beef burger!

Thank you!