Project 4

David Komnick Bebeta Hoax

October 21, 2022

Project 4 wants us to use neural networks to do a domain adaptation on the Cassava disease dataset. Diseased plants are supposed to be turned in to healthy ones. The architecture we used is published in the paper "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", the code is adapted from their github-repository.

1 Description of the data:

The diseased Cassava plants dataset contains images of healthy plants and diseased plants. The images contain a background (that should not be changed). Furthermore, the images have different shapes (horizontally and vertically taken images, images that are square-shaped). The images are unpaired, even though we have for each plant type examples for all classes.

We do not differentiate between the diseases for this task, even though the data set is distinguishing between several diseases. We define X as domain of healthy plants and Y as domain of diseased plants. For training the data we (of cause) normalize it. We reshape all images into a size of 5012×5012. Additionally, we scale the pixel values linearly down to a range from 0 to 1.F

2 Model

For solving this task, we want to use cyclic GANs. We use the architecture from . We use the jupyter-lab environment for training the network – the model is defined in normal .py files (for keeping the notebook clean).

Cyclic GANs can preform a domain adaptation form one domain to another with unpaired training samples. There for, 2 GANs are trained, one for a domain adaptation from X to Y and one for a domain adaptation from Y to X. Details for the training are given in the section "loss". The generator is not effecting the size of the data. So concatenating the generators should give the identity function (in optimal cases).

2.1 Architecture details

We define two GANs $G : X \to Y$ and $F : Y \to X$ for both directions of the domain adaptation. F and G have an identical architecture containing a discriminator and a generator.

The Discriminator is fully convolutional stack up (in out case 6 layers), the image is reduced to a size of 1×1.

The generator is a residual network, 3 convolutional layers are followed by a residual stackup. 3 Deconv-Layers are afterwards resizing the image to the original size. The residual stackup is very important to preserve the general image shape. The generator is able to preserve many parts of the input image without being learned. The image is not looing good at all - there are just some edges preserved. This makes training faster.

In general the activation is a variation of ReLU (the variation is taken for numerical stability.

2.2 Loss function

The loss function for the generator and discriminator from domain X to domain Y is described. The other iterator is trained equivalently.

Let $x \in X, y \in Y$. From each domain (X and Y) a batch is taken. The discriminator has one loss-part. The loss for $D_y(G_{X \to Y}(x))$ is calculated by a pixel-wise loss function.

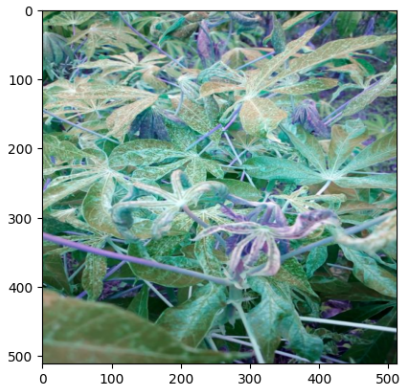

Figure 1: Plant with disease

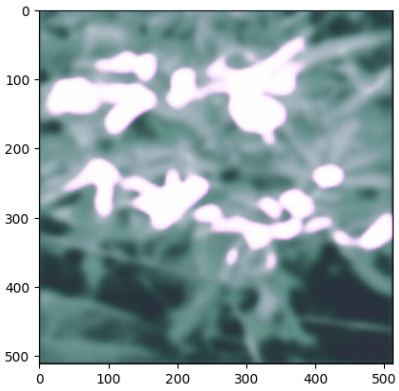

Figure 2: healthy plant

The loss for the generator is a combination of 3 parts. At first, the "standard generator training" for GANs is minimized ($\mathbb{E}[1 - D_y(G_{X\to Y}(x))]$. Additionally, we want the Discriminator the identity mapping is learned my minimizing $\mathbb{E}[1 - D_y(G_{X\to Y}(y))]$. As third part the cyclic-property is learned by minimizing $\mathbb{E}[1 - D_y(G_{X\to Y}(G_{Y\to X}(y)))]$.

The generator and discriminator trained rotationally.

3 Training

Because of the large size of the network, training needs a lot of time. Due to a lack of computational power, we trained our network for only one epoch. An epoch is limited by the amount of healthy plants. As optimizer we used ADAM.

The meta parameter (beside all meta parameter from the used layers, the optimizer and the general architecture itself) are the amount of layers we use in the generator for the convolutional part and the amount of residual layers. Since we are having a loss function combing 3 separate losses, an impotent meta parameter is how we weight the different parts of the loss function (with linear weights). Optimizing these parameter properly would require auto ML techniques most likely (which would consume much more time then available or reasonable).

3.1 Results and discussion

First of all it is hard to find an automatic measurement for testing how good the results are. There is no measurement for accuracy that we could apply since we do not know which image we want to generate. We could test how good the networks are preforming on as identity function or if we stack then together. But both times we do not measure "if any diseased parts are replaced with healthy parts" but that is the main goal of the network. Image 1 shows a diseased plant and image 2 shows the healthy version of it according to the GAN. We directly see, that some lighter parts are mapped to complete white parts and the image has far less details. But still we can see the general shape of the leaves and we would agree, that the image is not looking to diseased (even though very blurry). The background is transformed to a very dark color.

Taking a closer look on the leave we see in the bottom left corner (transferable to other leaves): The cursed parts of the leave are more likely to be mapped to a dark color, some diseased parts of this leave are mapped to the "default green leave color". Most likely the image will look much better after more training epochs.

3.2 remark

All in all it is very amazing that even with this little amount of training we achieve promising results - for being able to generate high resolution images more training time is needed (as already mentioned). Looking at the results from the paper the main issue of the network seems to be differentiating between background and actual image.