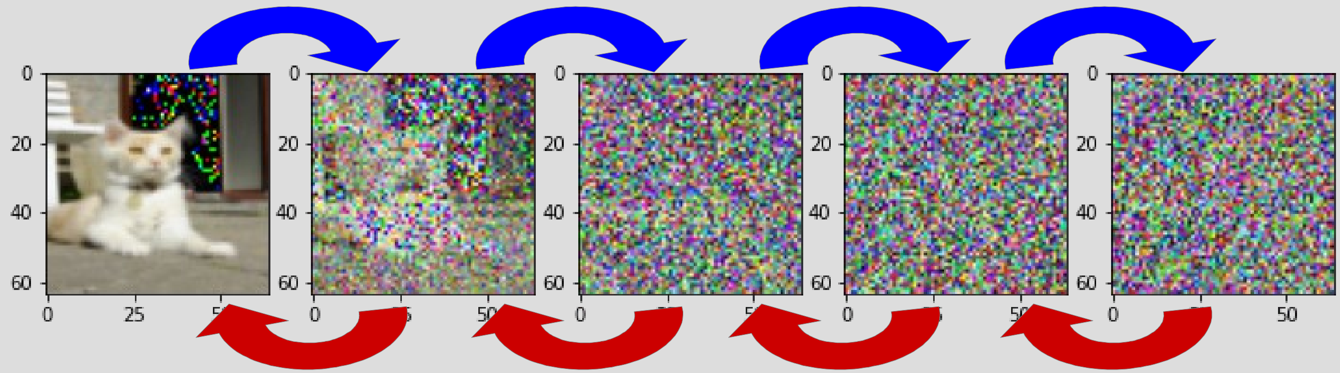
Diffusion Model Project

By Miguel Pinel Martínez

Deep Learning 2022

General overview

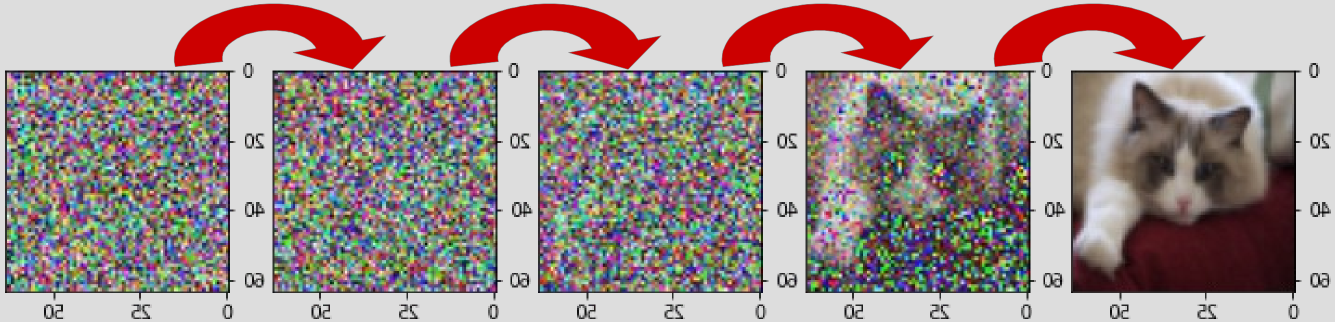
Forward diffusion process



Reverse diffusion process

Objective:

Start with random noise

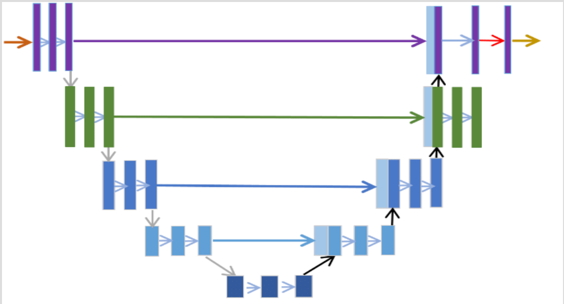


End with an original image

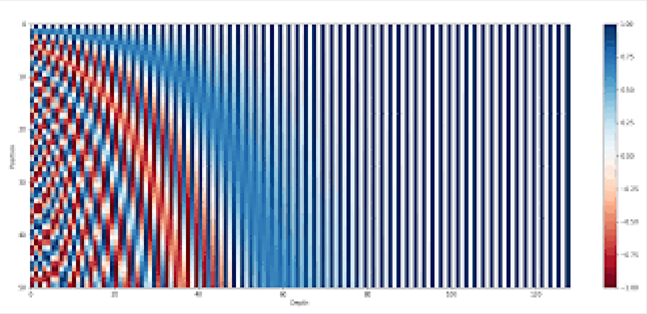
Main Components

Noise schedule

Denoising predictor
(neural network)



Timestep Embedding



Noise schedule

$$q(x_t|x_{t-1}) = N(x_t; \mu = \sqrt{1-\beta_t}x_{t-1}, \Sigma_t = \beta_t I)$$

New Image

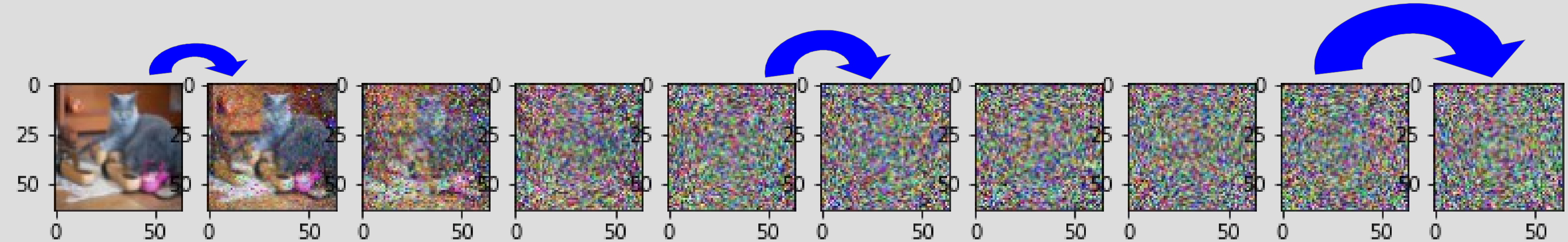
Previous Image

Noise parameter

What does the noise schedule do?
It increase the noise parameter in each timestep. We will use a linear one.

Low increase in noise

High increase in noise



Noise schedule optimization I

Can we calculate everything at one?

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

Reparameterization trick:

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_{t-1} = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

Notation: $\alpha_t = 1 - \beta_t$ $\bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$

Alphas

Cumulative Alphas

Noise schedule optimization II

$$x_t = \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} = ... = \sqrt{\bar{\alpha}} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$

So the final formula for calculating a timestep t:

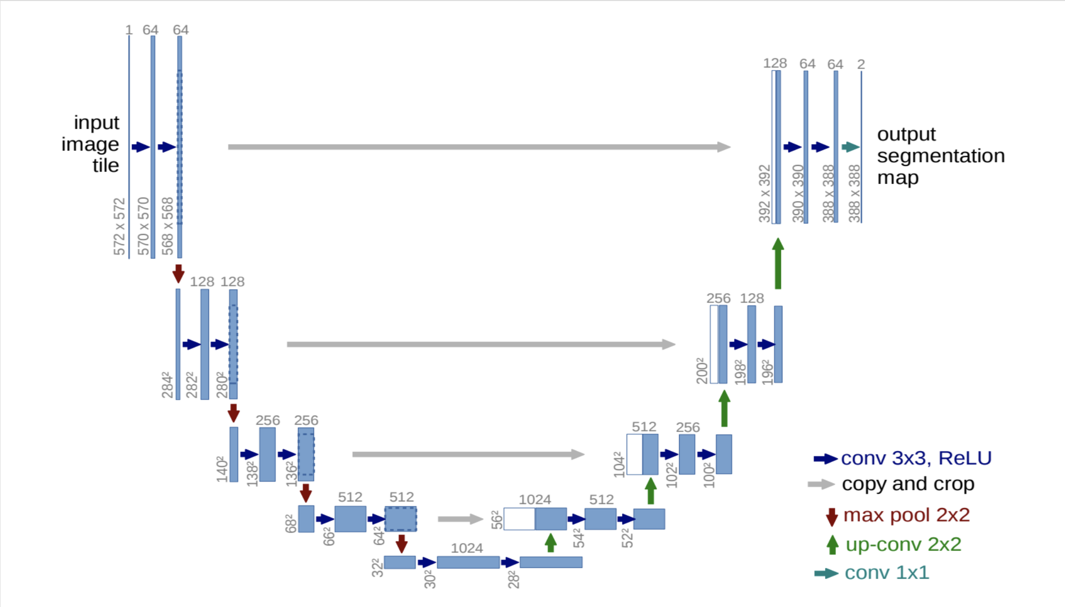
$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

Now we can calculate directly the noise version of an image in any timestep without needing to calculate all the intermediate steps

Denoise Predictor

We will need to learn the reverse diffusion process and for this we will use a neural network.

As we are working with images we will use a modified version of the U-Net architecture for this task



U-Net architecture

From: U-Net: Convolutional Networks for Biomedical Image Segmentation 2015

Loss function

Learnable funtion $\rightarrow$ $p_\theta(x_{t-1}|x_t) \approx q(x_{t-1}|x_t)$ $\leftarrow$ Objective function

This loss funtion is imposible to learn $Loss = -logp_\theta(x_0)$

We can stablish a lower bound to our loss function

$$-logp_\theta(x_0) \leq E_q(\underbrace{D_{KL}(q(x_T|x_0)||p_\theta(x_T))}_{\text{Constant term}}) + \Sigma_{t=2}^{T}\underbrace{D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))}_{\text{Stepwise denoising term}} \underbrace{- logp_\theta(x_0|x_1)}_{\text{Reconstruction term}}$$

Simplified Loss Function

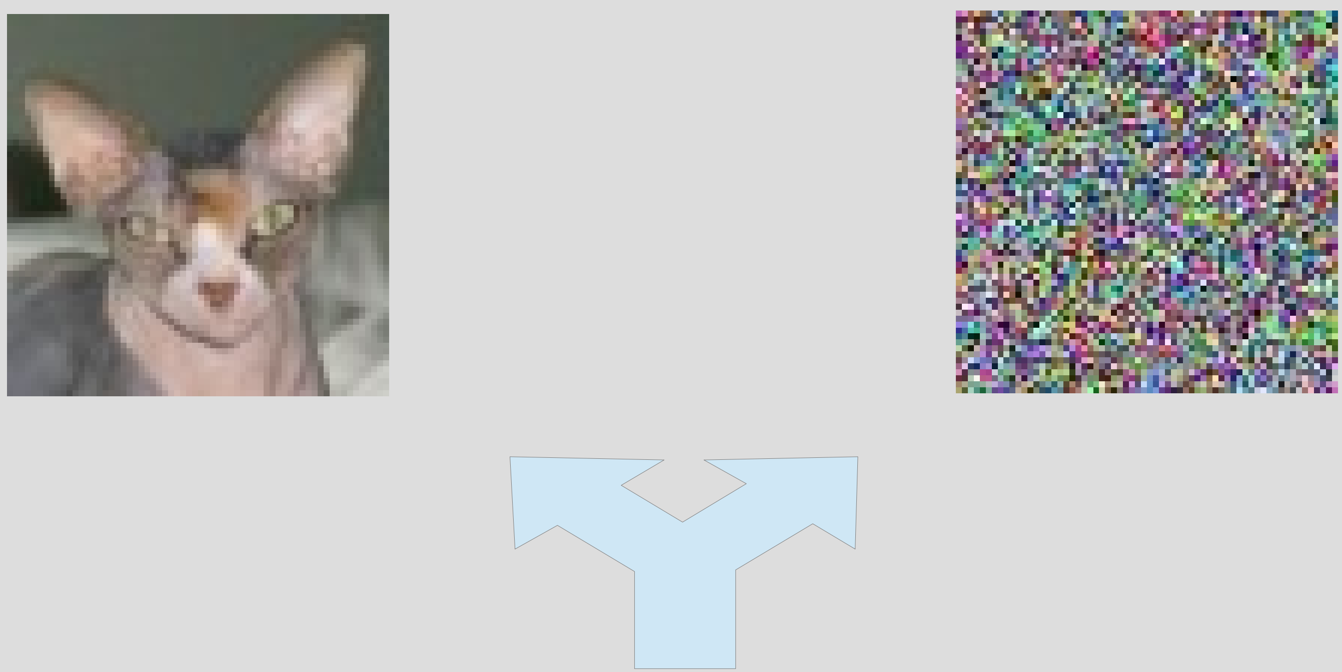
This is the most simplified loss funtion we can use

$$L_{simple} = E_{t,x_0,\epsilon}[||\epsilon - \epsilon_\theta(x_t, t)||^2]$$

In this loss function we don't calculate the denoise image, instead we predict the noise that have been added and by substracting it we obtain the denoise image

Time Embedding I

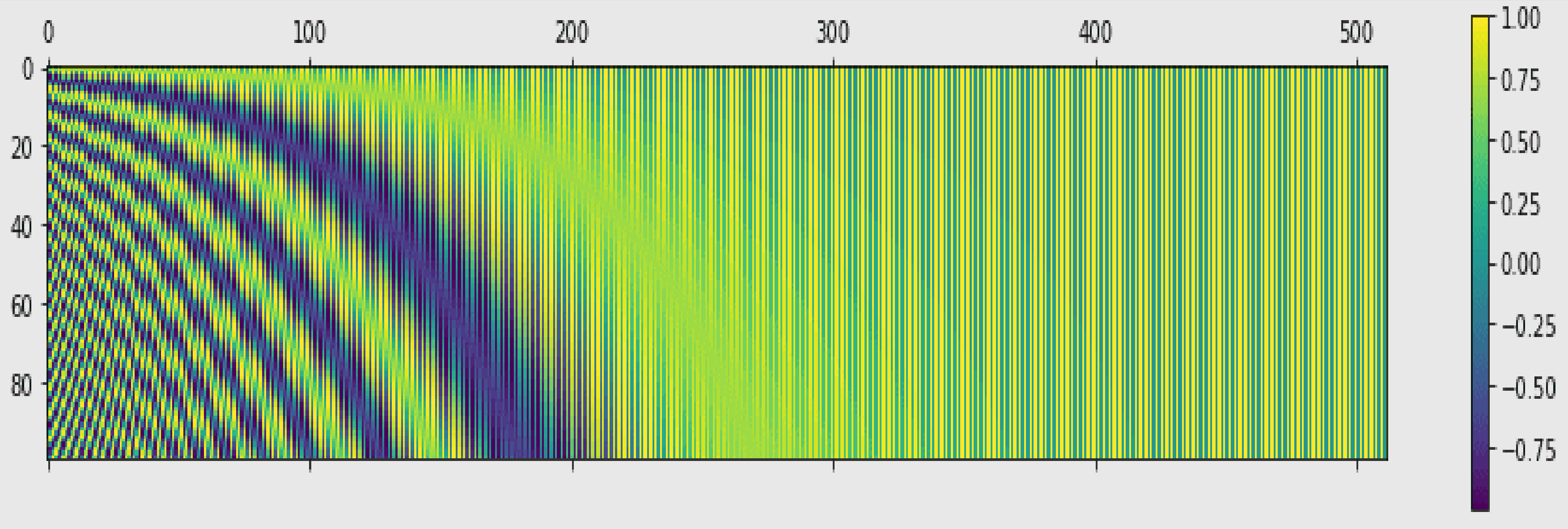
Our neural network will need a way of knowing how noisy is the image in which it is working



Both are equal for our neural network

Time Embedding II

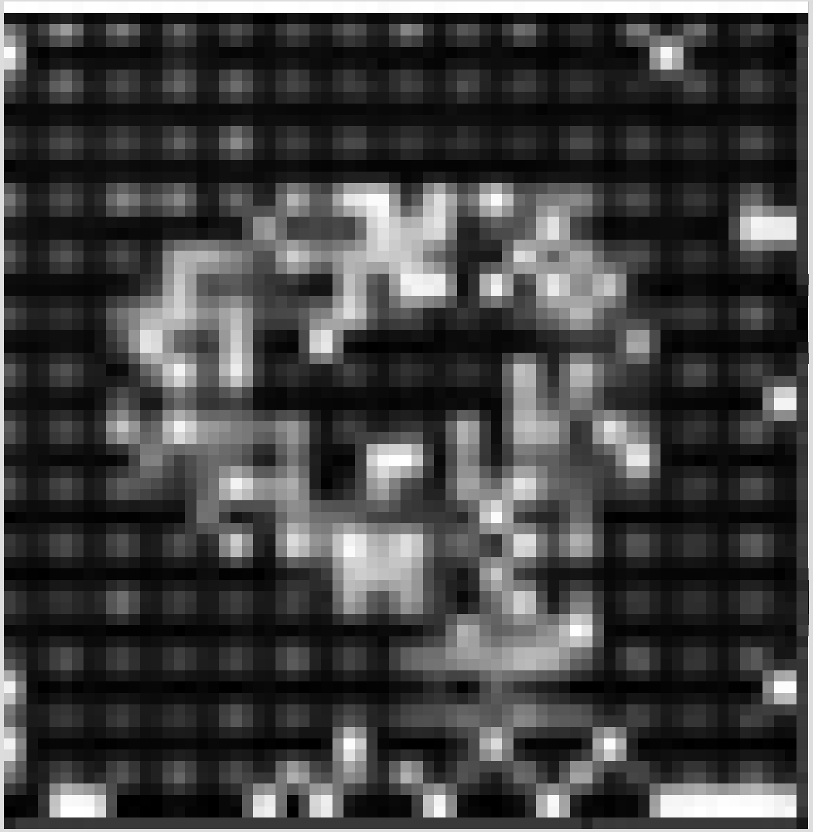
We will add a time embedding to our image in the same way as the postional encoding vectors in Transformers.



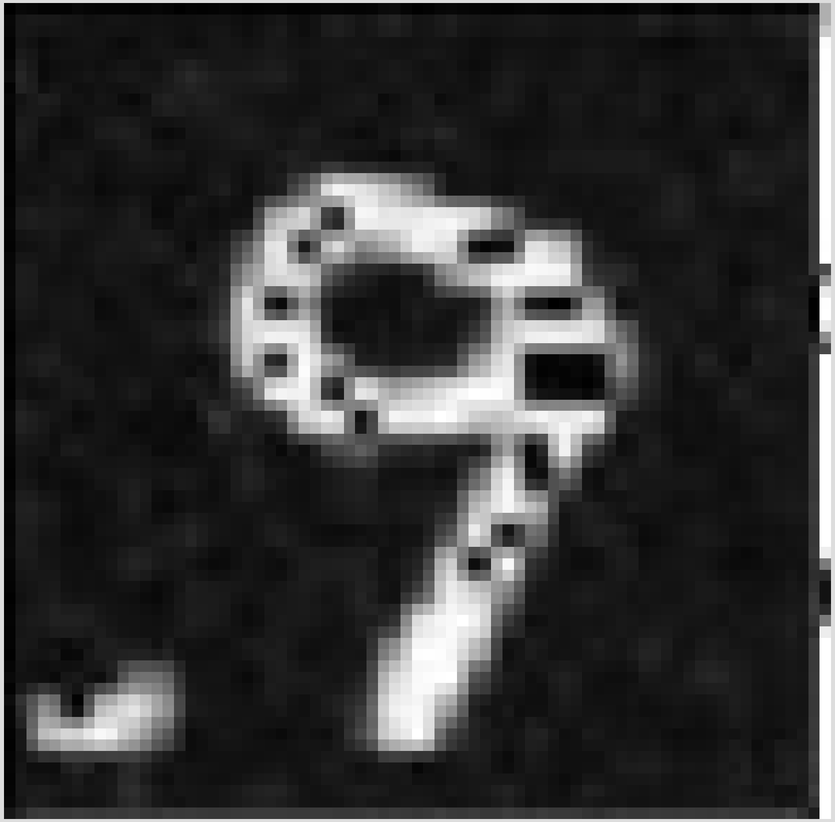
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Comparating with and without attention

Without attention

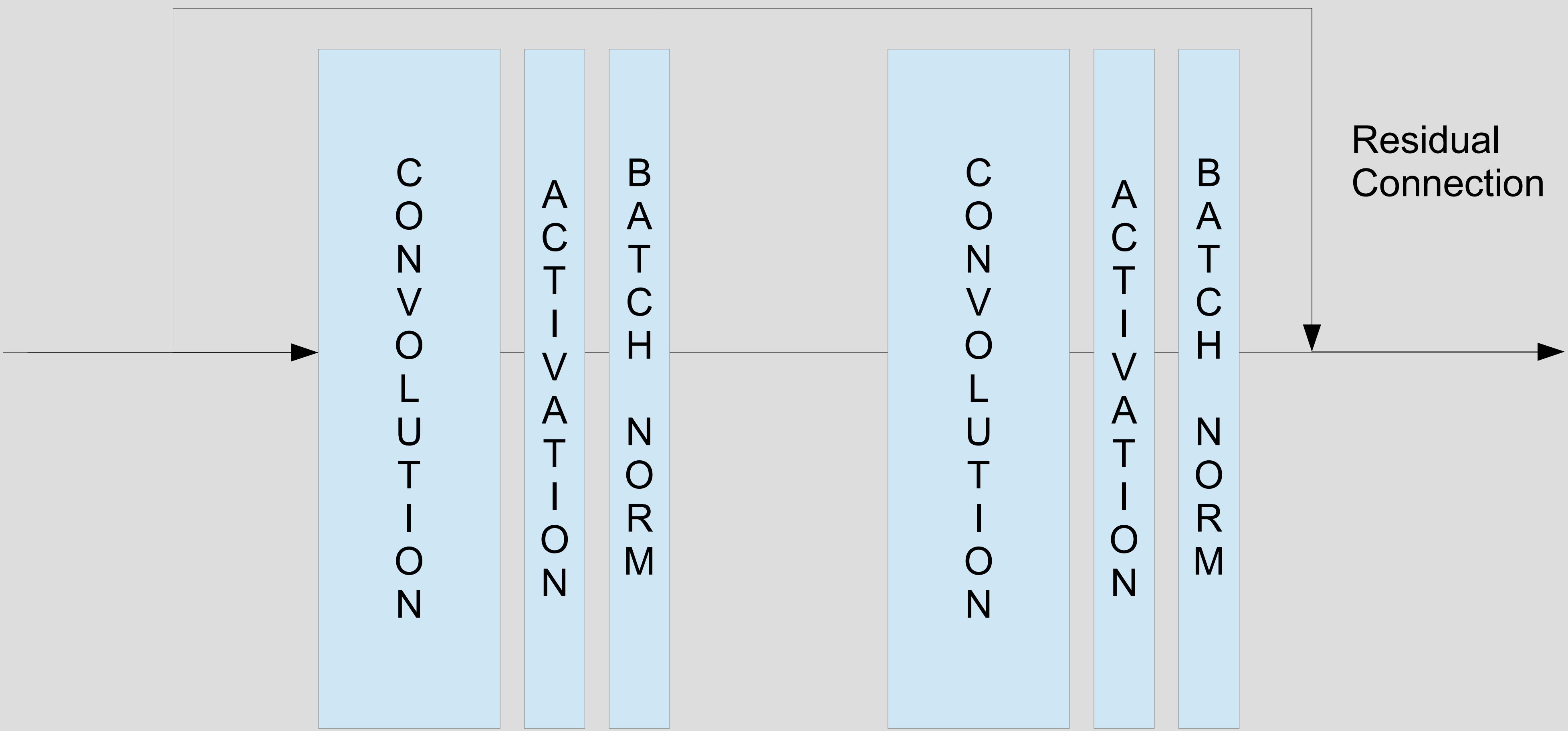


With attention



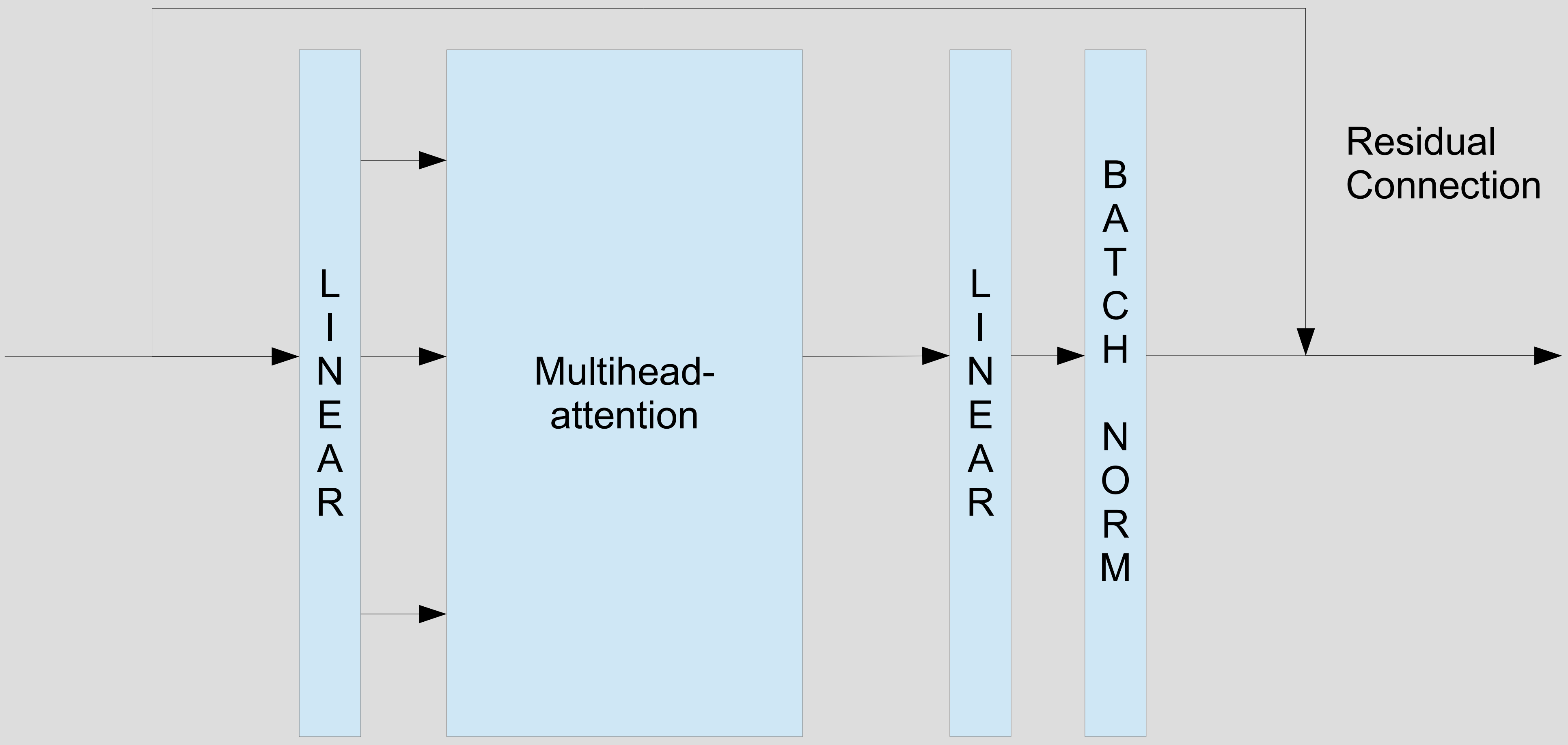
Neural Network Architecture I
Main Blocks

Residual Block



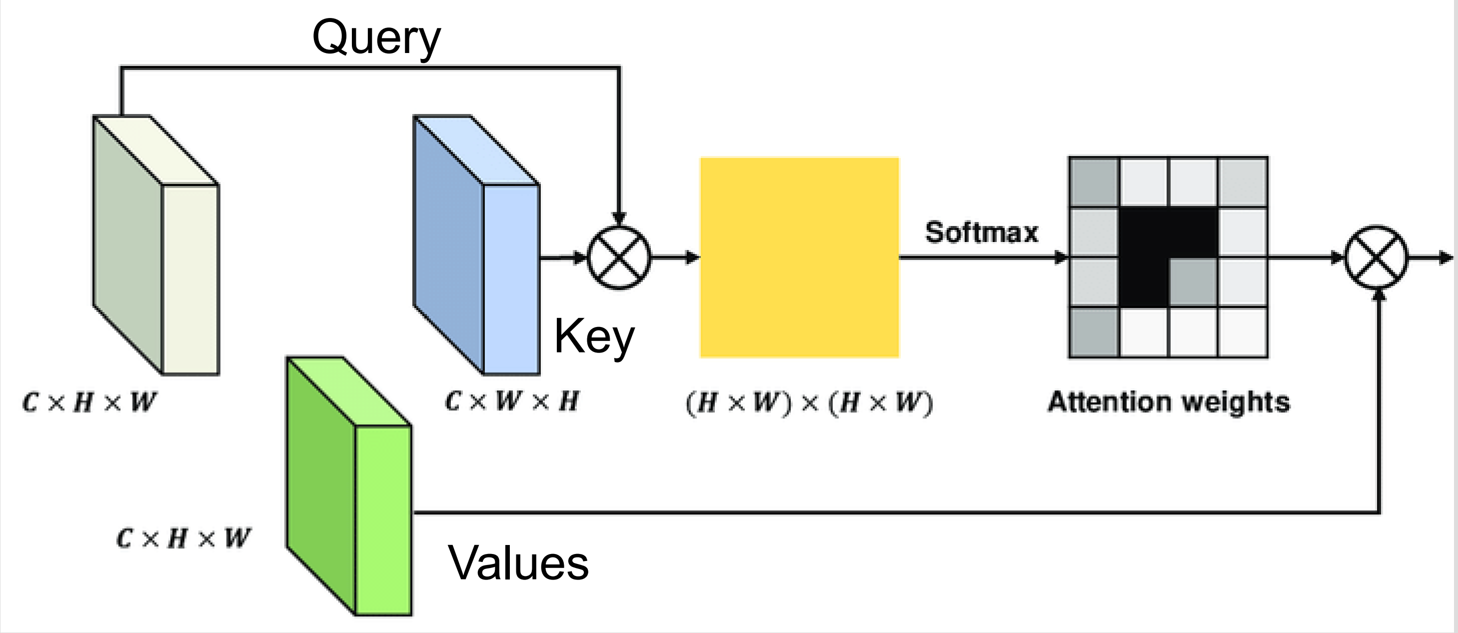
Neural Network Architecture II
Main Blocks

Attention Block

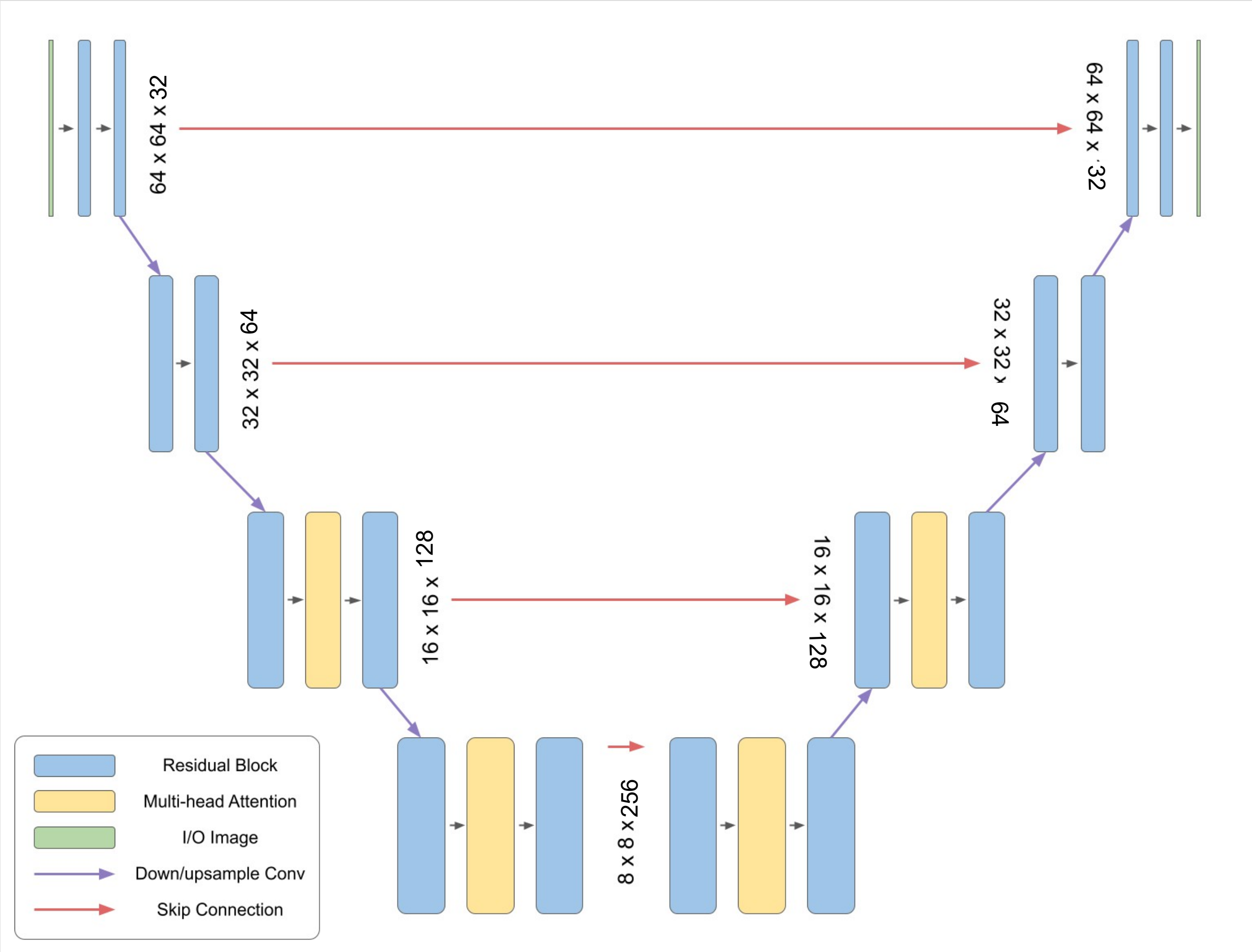


Neural Network Architecture III
Main Blocks

MultiheadAttention



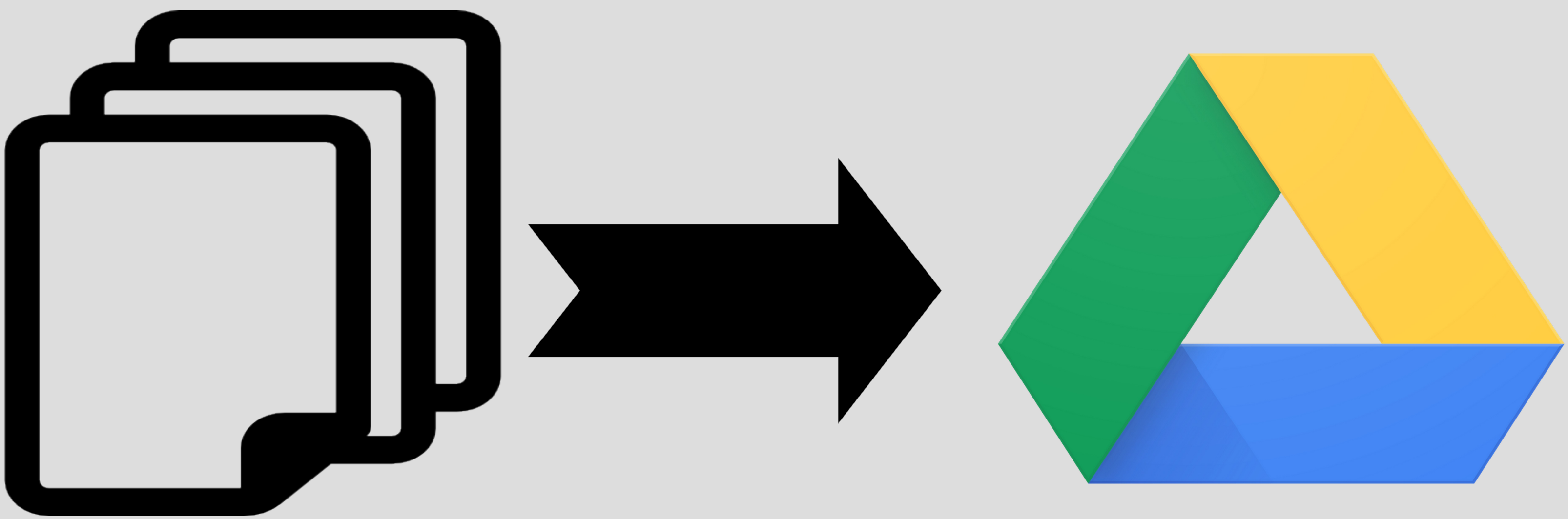
Neural Network Architecture IV



Architecture of the MinImagen implementation

Dataset I

First idea: Can we create our own dataset?

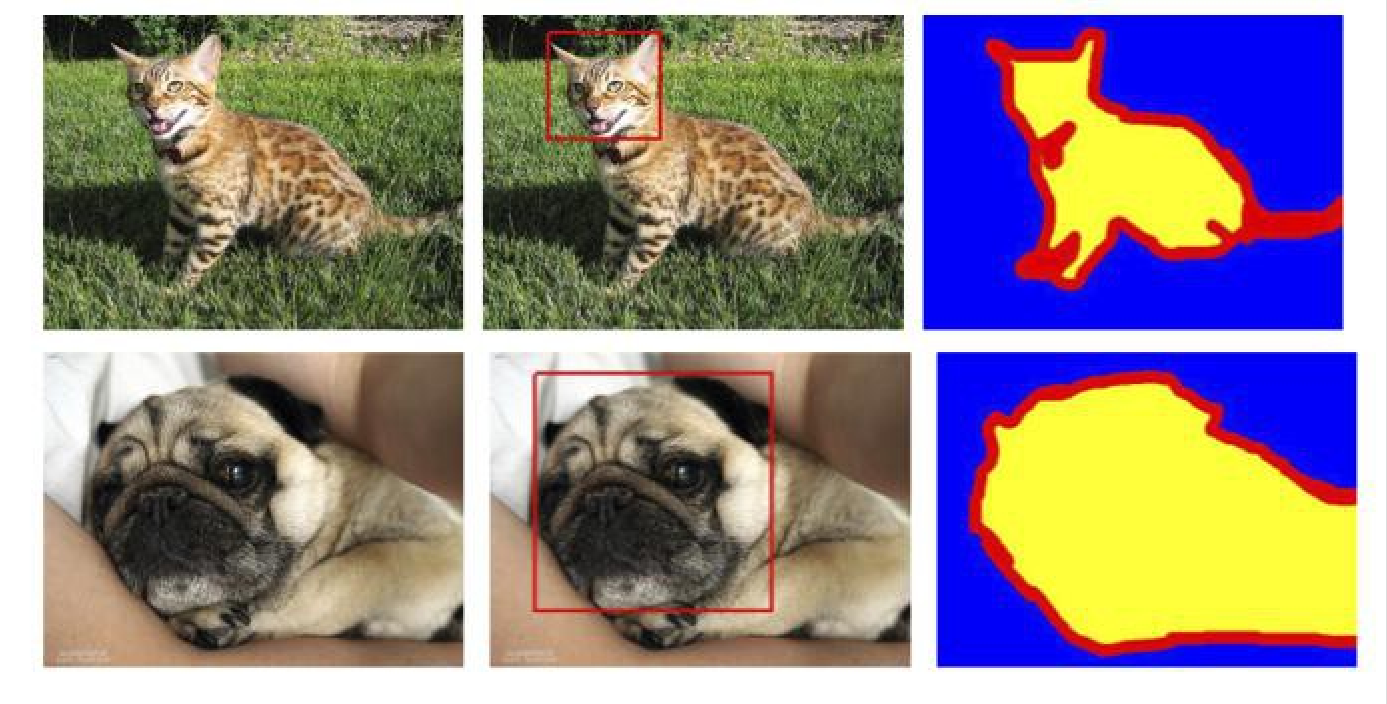


Main problem: Optimization

The datasets already incorporated in pytorch are highly optimized and this is a heavely consuming task.

Dataset II

So we will use the Oxford-IIIT Pet dataset already integrated in pytorch

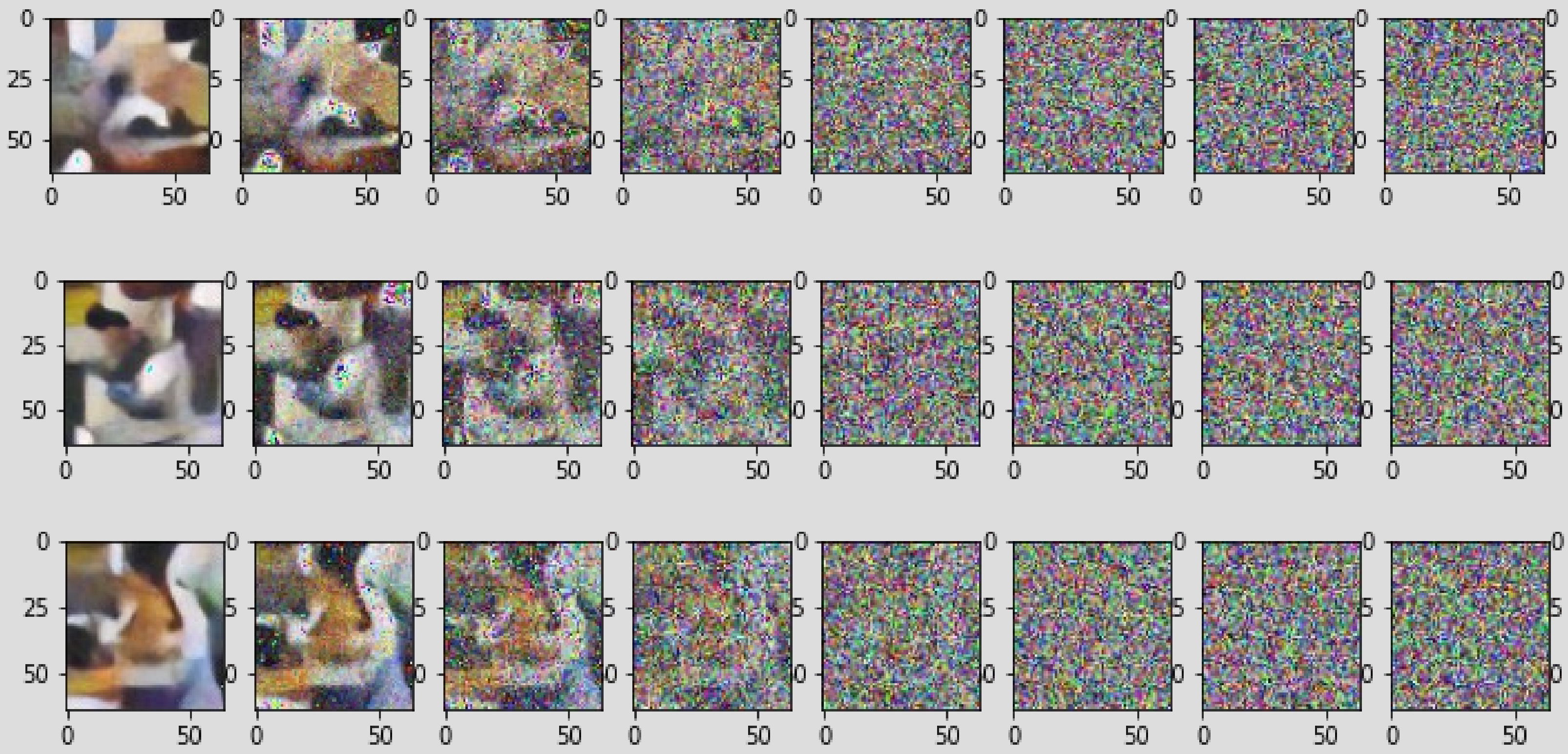


Main problem: Size of the dataset

This dataset includes 37 pets classes and 200 image for each class, but as we only want similar images we will not use all classes

Results of the model I

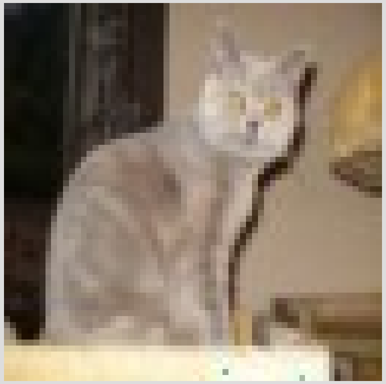
After all of this, we end up with a model of 12.952.451 parameters that have been train for 600 epochs or around 8 hours.
This are some of the images that the model has generate.



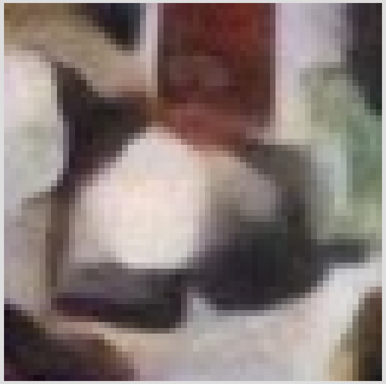
Results of the model II

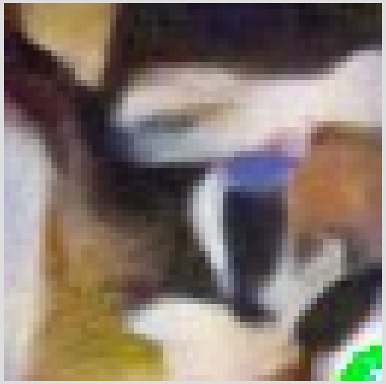
Let's compare some original images with the ones of the model

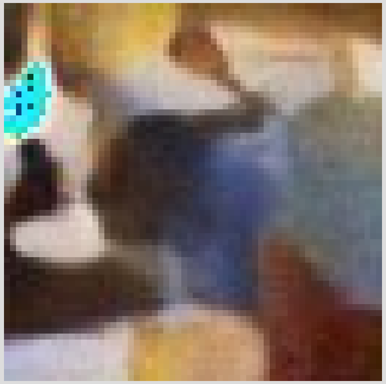
Original



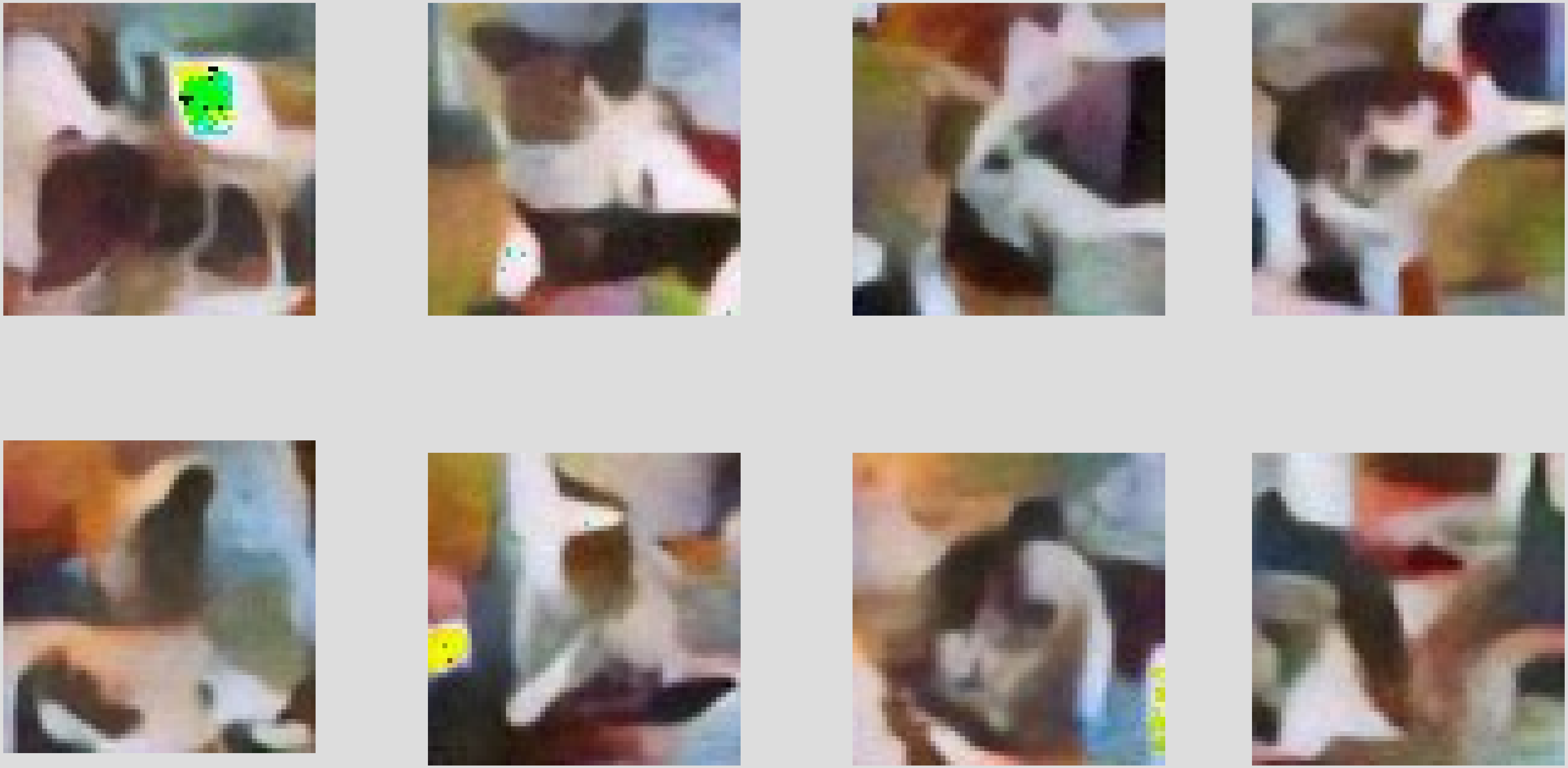
Artificial







Results of the model III



Results of the model IV

References

- Denoising Diffusion Probabilistic Models, 2020 , [Link]
- Improved Denoising Diffusion Probabilistic Models, 2021, [Link]
- Attention Is All You Need, 2017, [Link]
- Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding, 2022 [Link]
- U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015, [Link]