Generative Modeling by Estimating Gradients of the Data Distribution

Haladó Deep Learning Laboratórium

Furuglyás Kristóf, 2021. 05. 22

Tartalom

- Cikk ismertetése
- Reprodukálási célok
- Hálózat felépítése
- Tanulási fázisok
- Eredmények

Cikk Ismertetése

- GAN helyett adateloszláson alapul
- Stein-score: $\nabla_x \log p(x)$
- Loss: $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$

$$\ell(\boldsymbol{\theta};\sigma) \triangleq \frac{1}{2}\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\mathbb{E}_{\tilde{\mathbf{x}}\sim\mathcal{N}(\mathbf{x},\sigma^2 I)}\left[\left\|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}},\sigma)+\frac{\tilde{\mathbf{x}}-\mathbf{x}}{\sigma^2}\right\|_2^2\right]$$

- Alacsony dimenziószám, ritka térrészek

Cikk Ismertetése

- Langevin dinamika:

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\epsilon}{2}\nabla_{\mathbf{x}} \log p(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\epsilon}\, \mathbf{z}_t$$

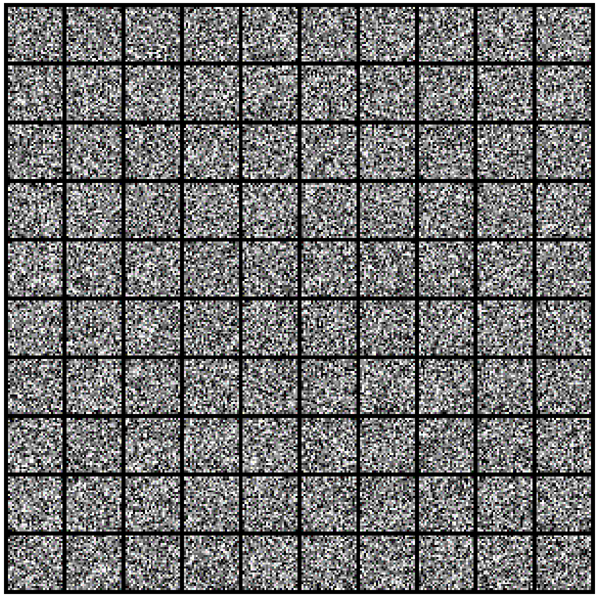
Algorithm 1 Annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T.$

Initialize $\tilde{\mathbf{x}}_0$
for $i \leftarrow 1$ to L **do**
  $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2/\sigma_L^2$ $\triangleright$ α_i is the step size.
  for $t \leftarrow 1$ to T **do**
    Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
    $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2}\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\, \mathbf{z}_t$
  end for
  $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
end for
return $\tilde{\mathbf{x}}_T$

Reprodukálási célok:

- Implementálni az algoritmust
- Betanítani az MNIST adatbázisra
- Kíhűlő Langevin dinamikával az alábbihoz hasonló számok generálása (cikk eredményei):



Hálózat

- Cikk: 4-cascaded RefineNet (U-Net)
- Saját:

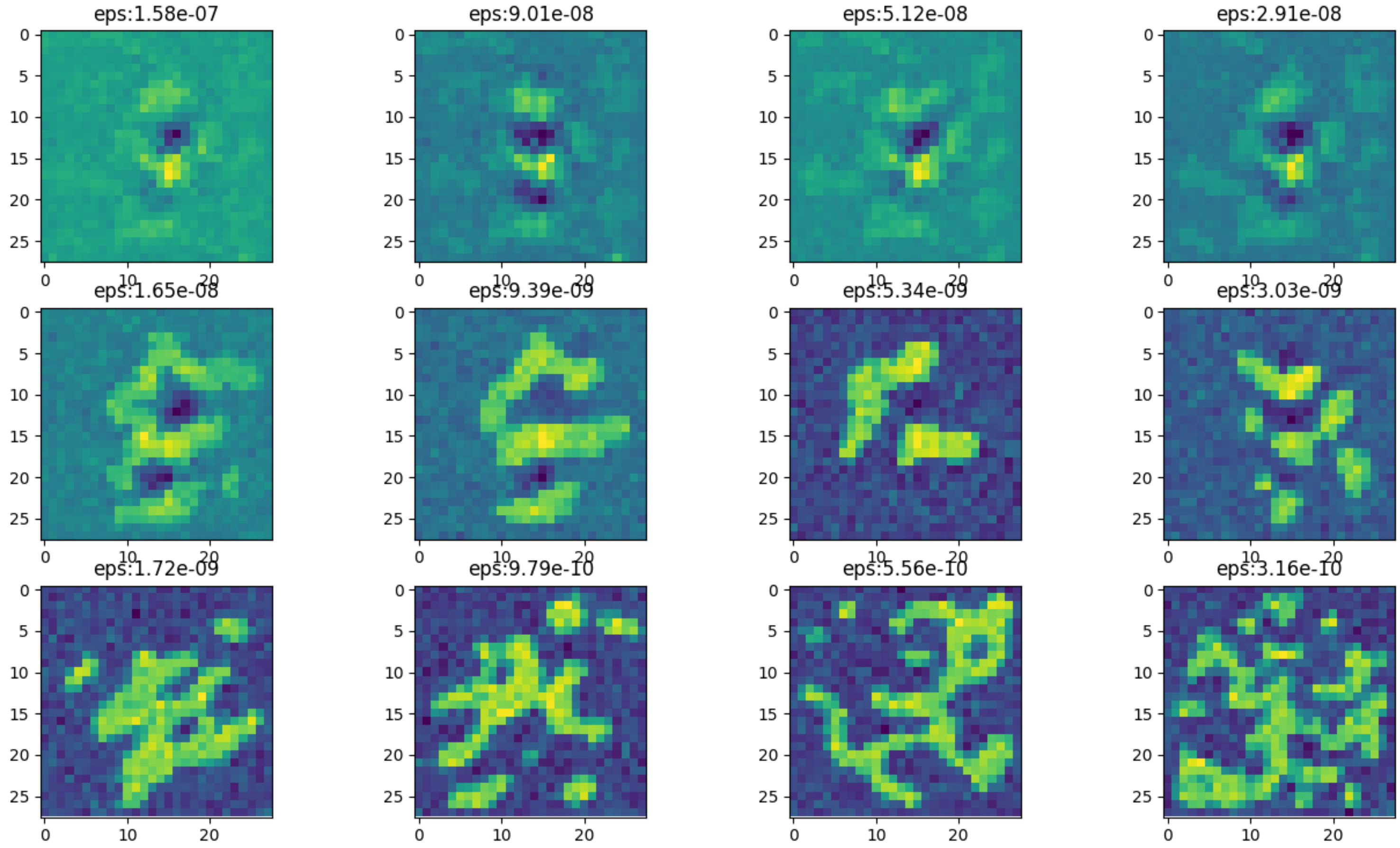
```python
class Score(nn.Module):
    def __init__(self, config):
        super().__init__()
        self.config = config
        nef = config.model.nef
        self.u_net = nn.Sequential(
            nn.Conv2d(config.data.channels, nef, 4, stride=2, padding=1),
            nn.GroupNorm(4, nef),
            nn.ELU(),
            nn.Conv2d(nef, nef * 2, 4, stride=2, padding=1),
            nn.GroupNorm(4, nef * 2),
            nn.ELU(),
            nn.Conv2d(nef * 2, nef * 4, 5, stride=1, padding=0),
            nn.GroupNorm(4, nef * 4),
            nn.ELU(),
            nn.ConvTranspose2d(nef * 4, nef * 2, 5, stride=1, padding=0),
            nn.GroupNorm(4, nef * 2),
            nn.ELU(),
            nn.ConvTranspose2d(nef * 2, nef, 4, stride=2, padding=1),
            nn.GroupNorm(4, nef),
            nn.ELU(),
            nn.ConvTranspose2d(nef, config.data.channels, 4, stride=2, padding=1),
            nn.ELU()
        )
        self.fc = nn.Sequential(
            nn.Linear(config.data.channels * config.data.image_size * config.data.image_size, 1024),
            nn.LayerNorm(1024),
            nn.ELU(),
            nn.Linear(1024, config.data.channels * config.data.image_size * config.data.image_size)
        )

        #self.layers = [l for l in self.u_net] + [l for l in self.fc]

    def forward(self, x):
        if x.is_cuda and self.config.training.ngpu > 1:
            score = nn.parallel.data_parallel(
                self.u_net, x, list(range(self.config.training.ngpu)))
        else:
            score = self.u_net(x)
        score = self.fc(score.view(x.shape[0], -1)).view(
            x.shape[0], self.config.data.channels, self.config.data.image_size, self.config.data.image_size)
        return score
```

Tanulási fázisok

- Paraméterek:
  - 10 különböző zajszint:
    $$\{\sigma_i\}_{i=1}^{10}, \sigma_1 = 10, \sigma_{10} = 0.01$$
  - T = 100 lépés, eps = 2e-5
- Egy hálót minden súllyal:
  - $\sigma_1 \to \sigma_{10}$
  - $\sigma_{10} \to \sigma_1$
- Minden súlyra külön háló

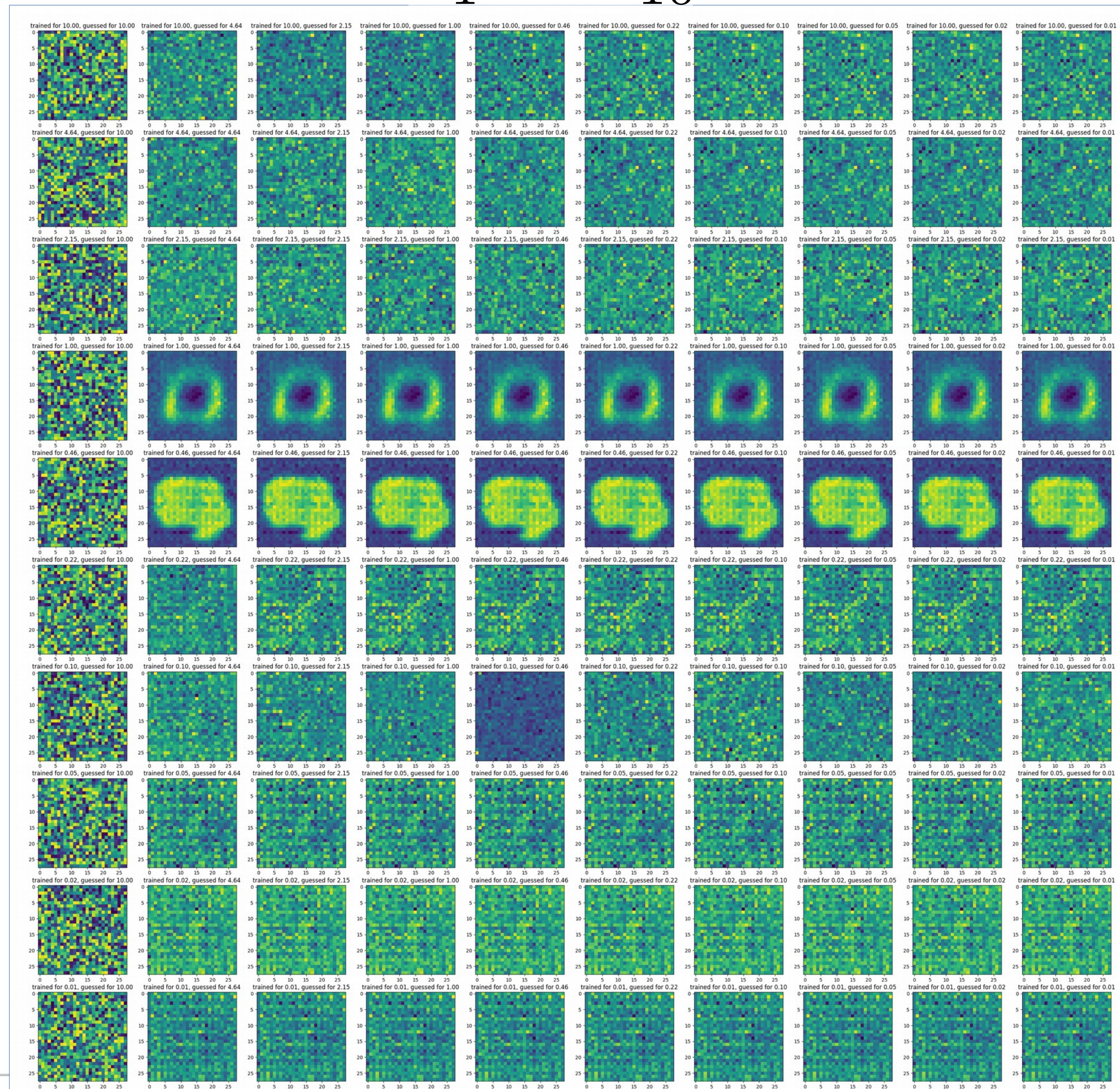
Eredmények

- Csak egy zajszint, különböző eps paraméterek:



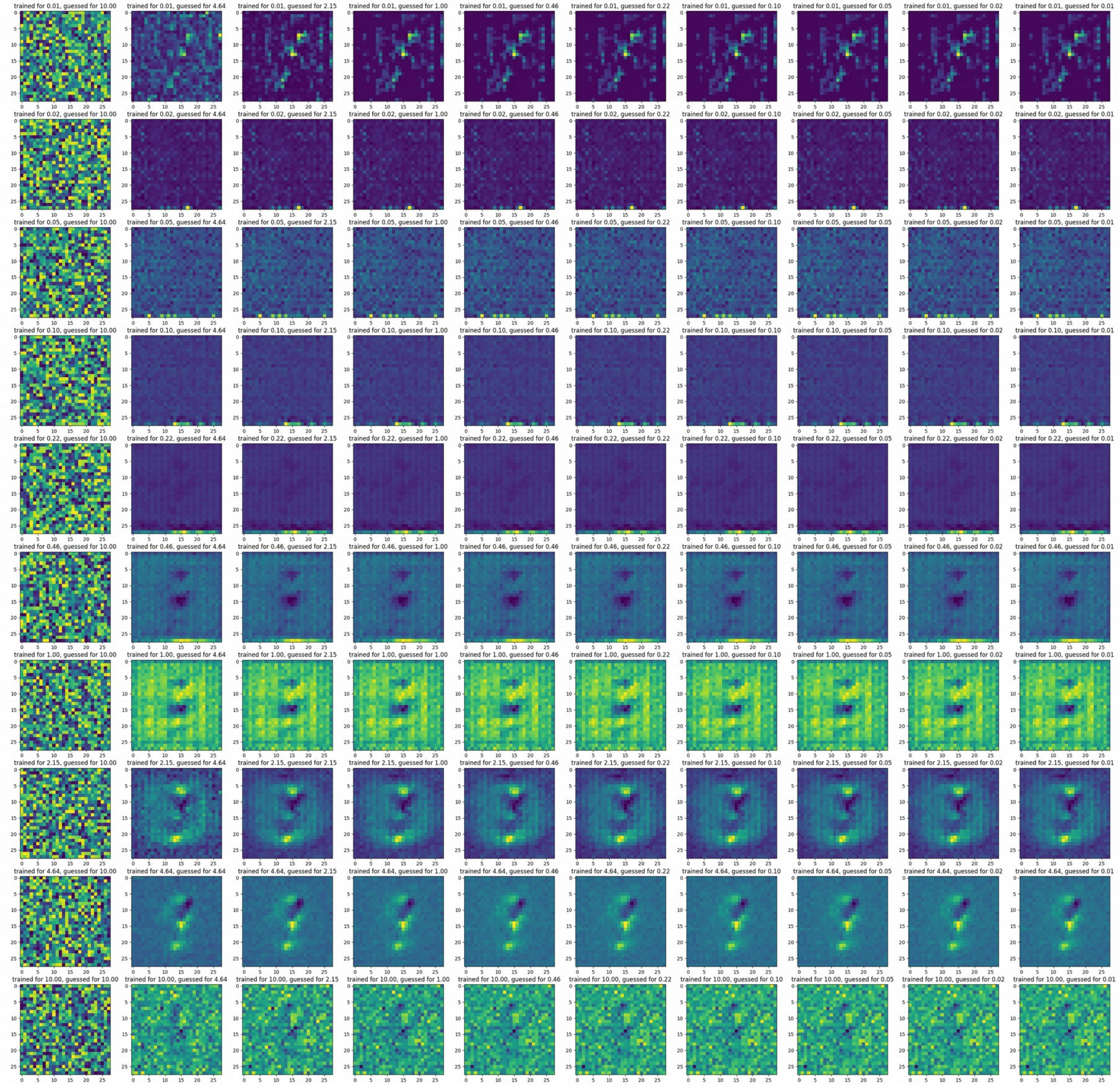
Eredmények

$\sigma_1 \to \sigma_{10}$



Eredmények

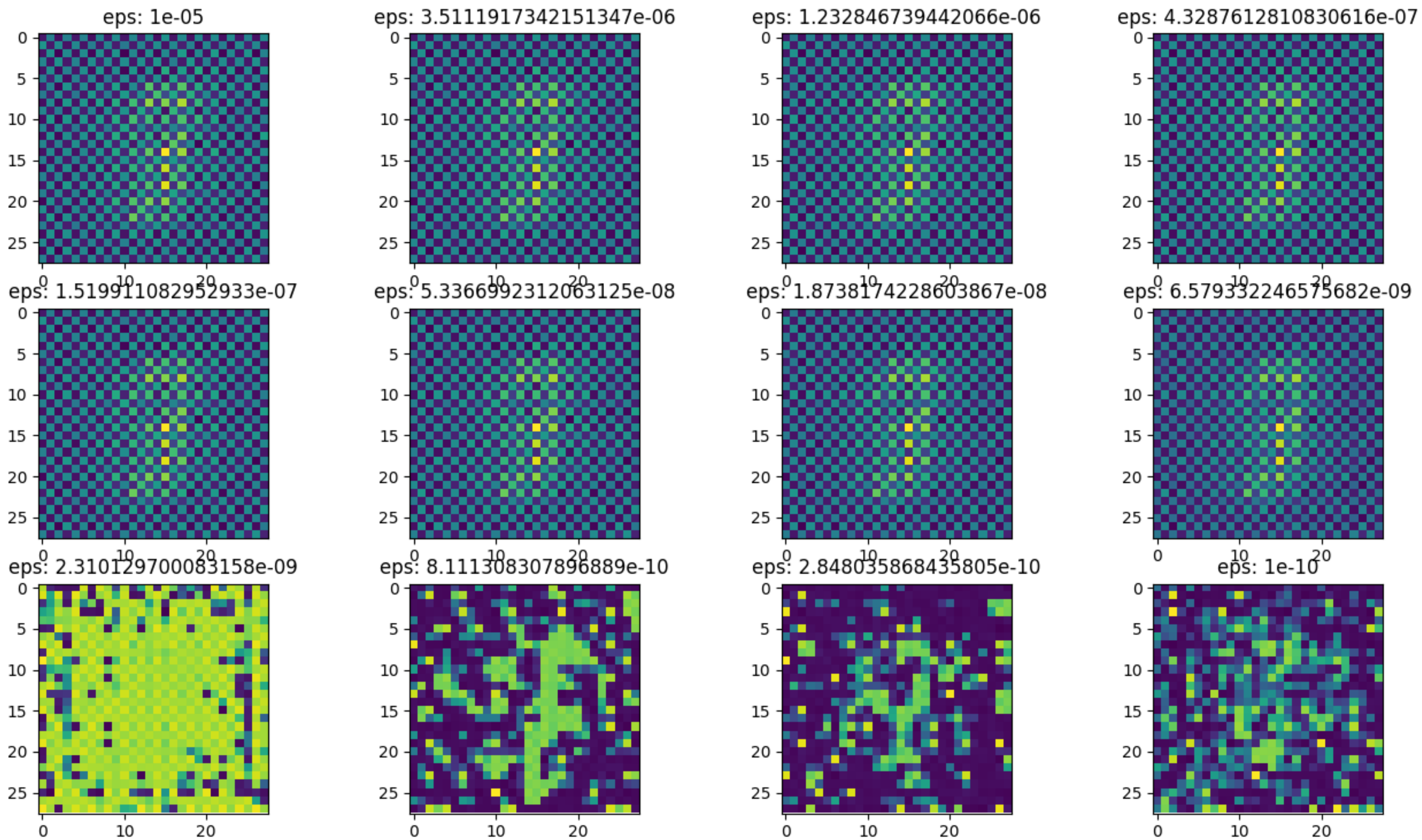
$\sigma_{10} \rightarrow \sigma_1$



Eredmények

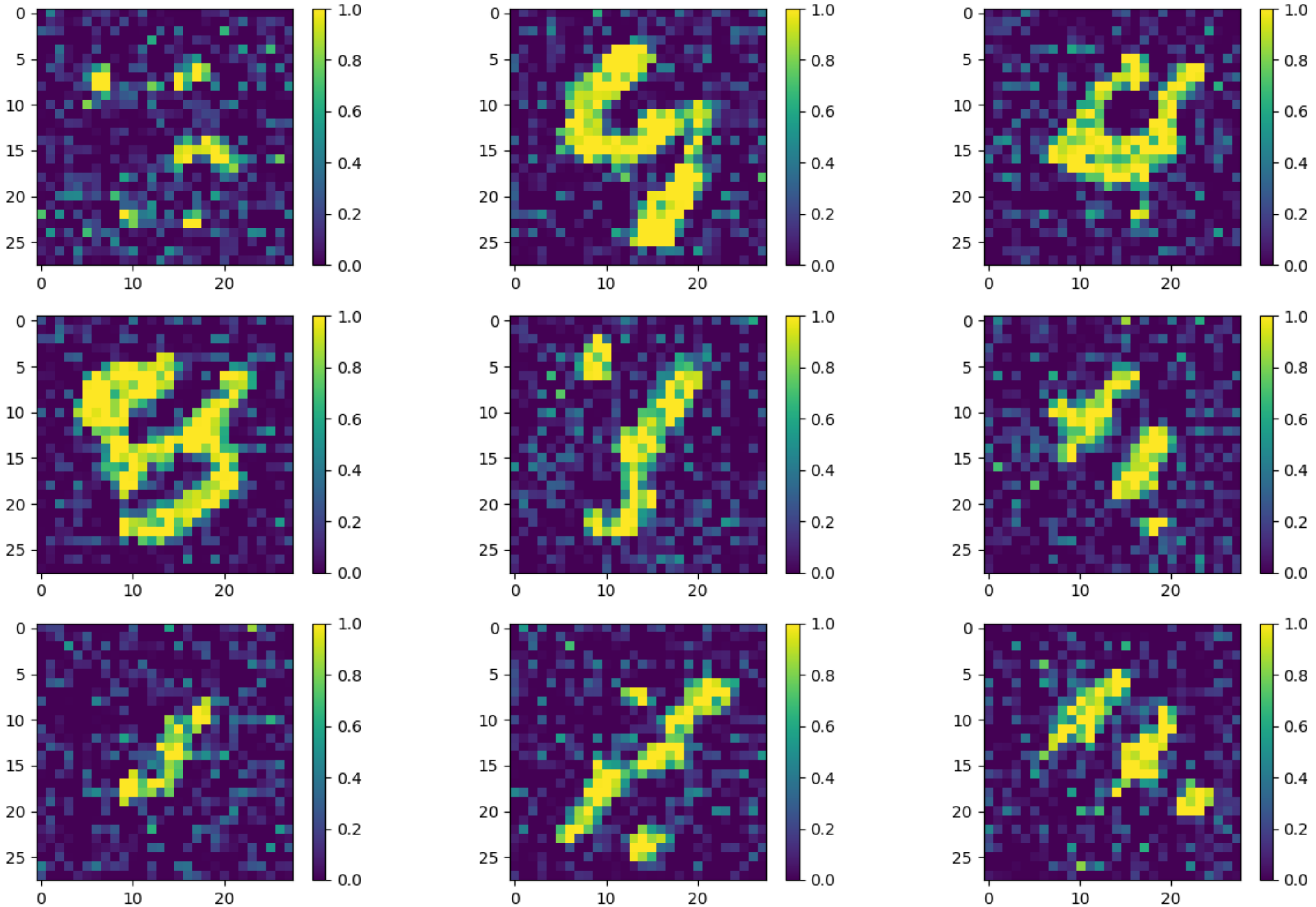
$$\sigma_{10} \rightarrow \sigma_1$$

- Több tanulási epoch, változó eps:



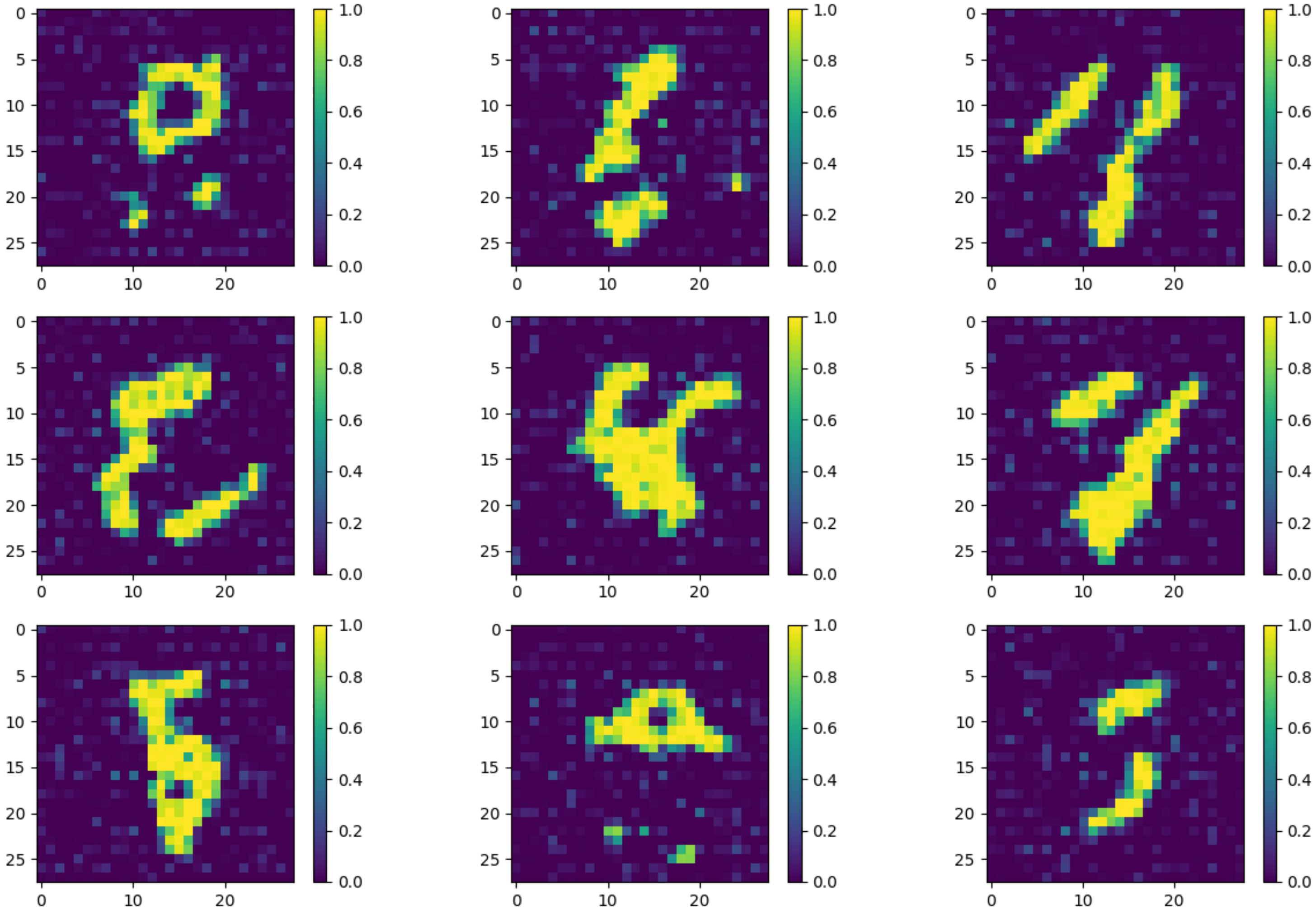
Eredmények

Külön hálókkal (100 epoch each)



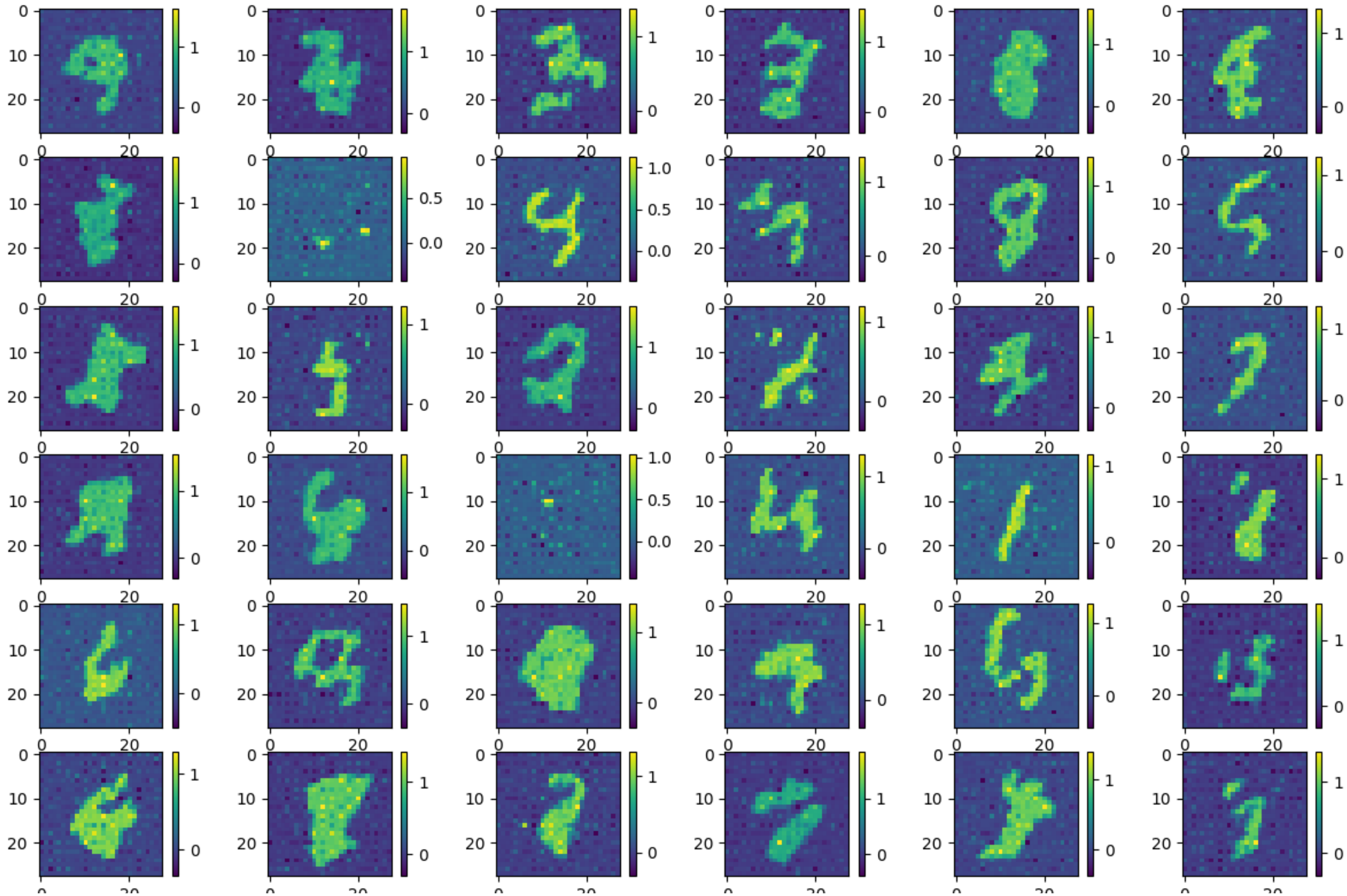
Eredmények

Külön hálókkal (400 epoch each)



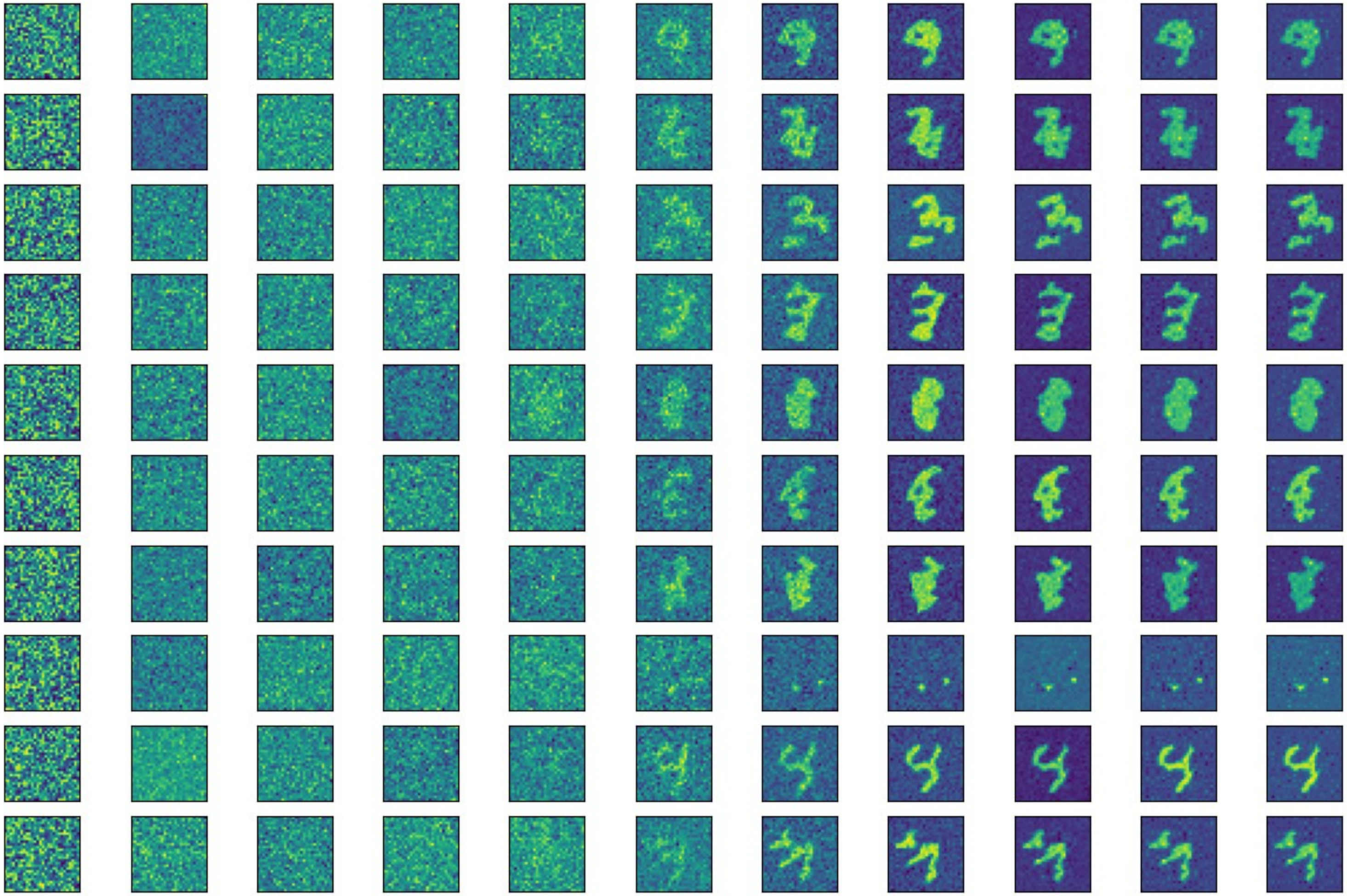
Eredmények

Külön hálókkal (400 epoch each)



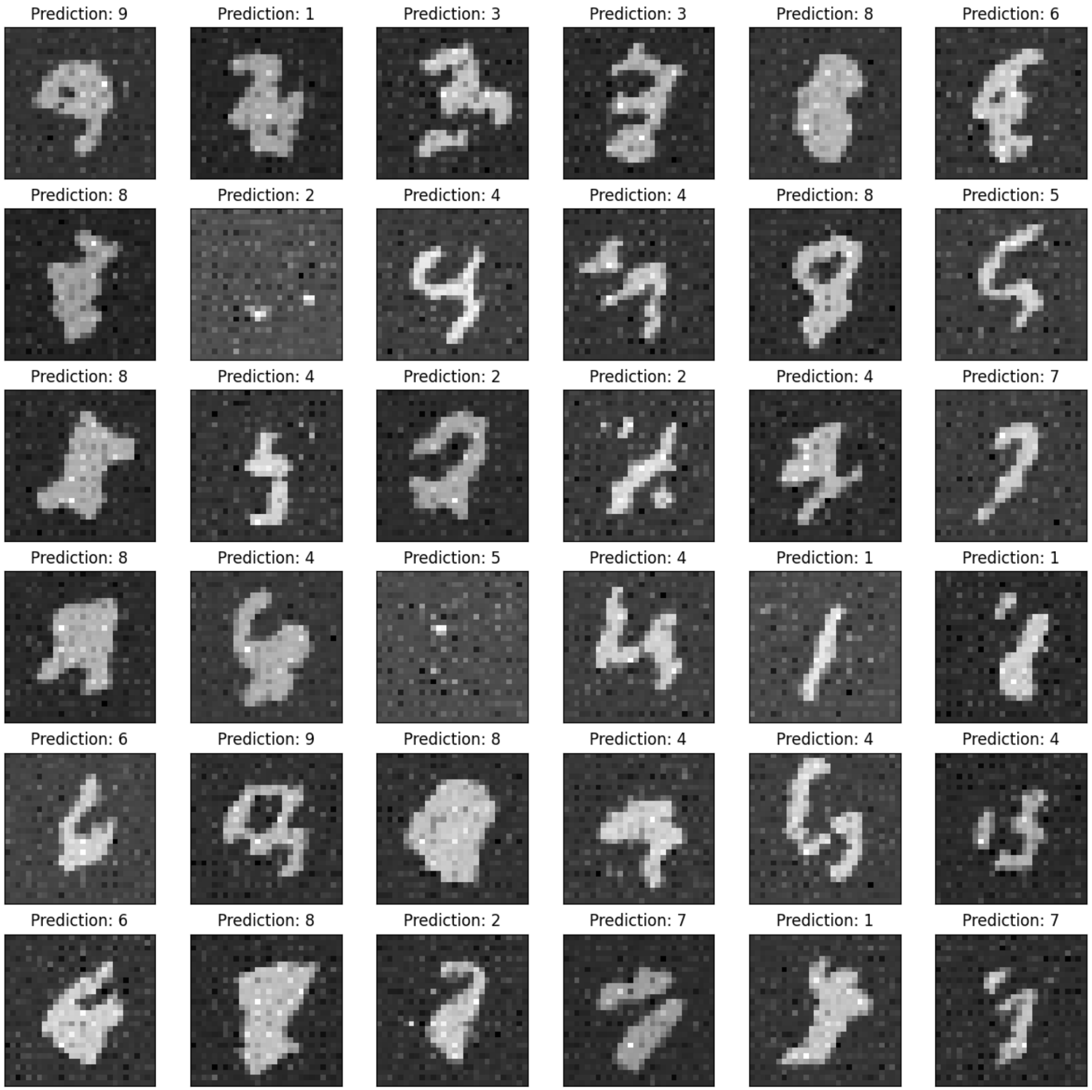
Eredmények

Külön hálókkal (400 epoch, fejlődés)



Eredmények

Külön hálókkal (400, preds)



Konklúzió

- Egyszerű, de több háló
- Hamis klaszeterek

- Javítások:
  - Komplexebb háló
  - Másik adatszet

Köszönöm a figyelmet!

Generative Modeling by Estimating Gradients of the Data Distribution

Haladó Deep Learning Laboratórium

Furuglyás Kristóf, 2021. 05. 22