

DeepDream

Luka Filipovic 83/2020

Jun 2024

Abstract

DeepDream (DD) je nova tehnologija koja funkcioniše kao kreativan pristup ureivanja slika korišćenjem reprezentacija CNN-a za proizvodnju slike poput snova tako što će iskoristiti prednosti i CNN-a i Inception-a i izgraditi san kroz implementaciju sloj po sloj. Kako dani prolaze, DD postaje široko korišćen u oblastima veštačke inteligencije (AI). Ovaj rad je prvi sistematski pregled DD. Fokusirali smo se na definiciju, važnost, pozadinu i aplikacije DD. Obrada prirodnog jezika (NLP), slike, video i audio su glavna polja u kojima se DD primenjuje. Konačno, neke zanimljive preporuke su navedene da služe istraživačima u budućnosti.

Sadržaj

1	Uvod	2
2	InceptionV3 CNN	2
3	InceptionV3 klasa	3
3.1	Konstruktor (__init__ metod)	3
3.2	Forward metod	3
4	DeepDream klasa	4
4.1	Konstruktor (__init__ metod)	4
4.2	Forward metod	4
4.3	get_required_layers metod	4
5	Opšta arhitektura	5
6	Slučajevi upotrebe	5
7	DeepDream funkcija	6
7.1	Inicijalno podešavanje	7
7.2	DeepDream iteracije	7
7.3	Čuvanje rezultata	8
7.4	Rezime	8
7.5	Upotreba	8

8	Podešavanje parametara	9
9	Podešavanje težina klasa	9
10	Treniranje i obučavanje modela	9
11	Primena modela na skup podataka	9
12	Rezultati	10
13	Literatura	11

1 Uvod

DeepDream (DD) je nova tehnologija koju su 2015. predstavili Mordvintsev i njegov tim u Guglu. Koristeći CNN, DD ima za cilj da poboljša obrasce slike pomoću robusnih AI algoritama. Inception i Deep CNN predstavljaju temeljne osnove dubokog sna. Google-ov Deep Dream je tipična primena za ovu tehniku. DD radi na poboljšanju slika poboljšavajući njihove vizuelne atribute. Vizuelizacija DD je korišćena da se utvrdi da li je CNN ispravno naučio prave karakteristike slike. Dakle, DD se stvara tako što se slika sve više unosi u mrežu gde prvi slojevi detektuju prve karakteristike niskog nivoa (tj. ivice). Zatim se pojavljuju karakteristike visokog nivoa (tj. lica i drveće), idu dublje u mrežu. Konačno, retki završni slojevi prikupljaju sve one za konfigurisanje kombinovanih efekata (npr strukture ili drveće).

2 InceptionV3 CNN

Inception-v3 je arhitektura konvolutivne neuronske mreže iz porodice Inception koja čini nekoliko poboljšanja uključujući korišćenje Label Smoothing, Faktorizovanih 7×7 konvolucija i upotrebu pomoćnog klasifikatora za širenje informacija o etiketi niže niz mrežu (zajedno sa upotrebom serije normalizacija za slojeve u bočnoj glavi).

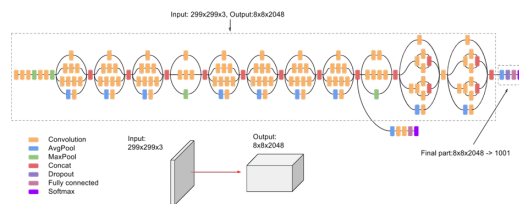


Figure 1: InceptionV3 CNN

3 InceptionV3 klasa

3.1 Konstruktor (`__init__` metod)

```
1 def __init__(self, num_classes=2, aux_logits=False, transform_input=
  False):
2     super(InceptionV3, self).__init__()
3     self.inception = models.inception_v3(pretrained=False, num_classes=
      num_classes, aux_logits=aux_logits, transform_input=
      transform_input)
```

- **Svrha:** Inicijalizovati InceptionV3 model.
- **Parametri:**
 - `num_classes`: Broj izlaznih klasa za konačni klasifikacioni sloj.
 - `aux_logits`: Da li treba koristiti pomoćne logite (logit je izlaz neuronske mreže pre nego što se primeni aktivaciona funkcija, podrazumevano je na `False`).
 - `transform_input`: Da li treba transformisati ulaz (podrazumevano na `False`).
- **Funkcionalnost:**
 - Poziva roditeljsku klasu konstruktor koristeći `super()`.
 - Inicijalizuje InceptionV3 model iz `torchvision.models`, sa podešenim parametrima. Model nije istreniran (`pretrained=False`), što znači da počinje sa nasumičnim težinama.

3.2 Forward metod

```
1 def forward(self, x):
2     return self.inception(x)
```

- **Svrha:** Definiše se forward prolaz za InceptionV3 model.
- **Parametri:**
 - `x`: Unosi tensor.
- **Funkcionalnost:** Propušta ulaz `x` kroz `inception` model i vraća izlaz.

4 DeepDream klasa

4.1 Konstruktor (`__init__` metod)

```
1 def __init__(self, model, layer_idx):  
2     super(DeepDream, self).__init__()  
3     self.features = self.get_required_layers(model, layer_idx)
```

- **Svrha:** Inicijalizovati DeepDream model.
- **Parametri:**
 - `model`: Instanca modela neuronske mreže (npr., InceptionV3).
 - `layer_idx`: Indeks sloja do kojeg se izdvajaju karakteristike.
- **Funkcionalnost:**
 - Poziva roditeljsku klasu konstruktor koristeći `super()`.
 - Izvlači potrebne slojeve iz `model` do `layer_idx` koristeći `self.get_required_layers`.

4.2 Forward metod

```
1 def forward(self, x):  
2     return self.features(x)
```

- **Svrha:** Definiše se forward prolaz za DeepDream model.
- **Parametri:**
 - `x`: Unosi tensor.
- **Funkcionalnost:** Prolazi unos `x` kroz izvadjene slojeve (`self.features`) i vraća izlaz

4.3 `get_required_layers` metod

```
1 def get_required_layers(self, model, layer_idx):  
2     if isinstance(model, InceptionV3):  
3         return nn.Sequential(*list(model.inception.children())[ :  
4                                 layer_idx+1])  
5     elif isinstance(model, nn.Module):  
6         return model  
7     else:  
8         raise ValueError("Unsupported model type")
```

- **Svrha:** Izvlači potrebne slojeve iz datog modela.
- **Parametri:**

- `model`: Instanca modela neuronske mreže.
- `layer_idx`: Indeks sloja do kojeg se izdvajaju karakteristike.

- **Funkcionalnost:**

- Proverava da li je model instanca `InceptionV3`.
- Ako jeste, izdvaja slojeve do `layer_idx` iz `inception` podmodula i pakuje ih u `nn.Sequential`.
- Inače, ako je model instanca `nn.Module` (druge vrste modela), vraća ceo model.
- A ako je model nepodrživ, onda izbacuje `ValueError`.

5 Opšta arhitektura

1. InceptionV3 model:

- Ovo je predefinisana arhitektura iz `torchvision.models` koja se može prilagoditi za određeni broj izlaznih klasa.
- Obično se koristi za zadatke klasifikacije slika.

2. DeepDream model:

- Ovo je prilagodjena neuronska mreža koja koristi osnovni model (npr., `InceptionV3`) i izdvaja karakteristike iz njega do određenog sloja.
- Korisno za vizuelizaciju unutrašnjih reprezentacija osnovnog modela, koji se obično koristi u tehnikama kao što je DeepDream.

6 Slučajevi upotrebe

- **InceptionV3:**

- Može se koristiti direktno za zadatke klasifikacije, sa brojem klasa određenim tokom inicijalizacije.

- **DeepDream:**

- Dizajniran da vizuelizuje i pojača obrasce koje je naučio osnovni model (`InceptionV3`).
- Može se koristiti da se razume koje karakteristike je model naučio generisanjem slika koje maksimizuju aktivaciju određenih slojeva.

Ova arhitektura kombinuje snagu unapred definisanog kompleksnog modela (`InceptionV3`) sa mehanizmom za ekstrakciju prilagodjenih karakteristika (`DeepDream`) kako bi se stvorio fleksibilan i pronicljiv alat za duboko učenje.

7 DeepDream funkcija

```
1 def deep_dream(image_tensor, model, layer_idx, iterations, lr,
2   octave_scale, output_path):
3     # Convert image tensor to nn.Parameter
4     img = nn.Parameter(image_tensor.to(device))
5
6     # Define the deep dream model
7     dream_model = DeepDream(model, layer_idx).to(device)
8
9     # Define the optimizer
10    optimizer = optim.Adam([img], lr=lr)
11
12    # DeepDream iterations
13    for i in range(iterations):
14        optimizer.zero_grad()
15        features = dream_model(img)
16        loss = features.norm()
17        loss.backward()
18        optimizer.step()
19
20        # Apply the octave scaling
21        img.data = img.data + octave_scale * img.grad.data
22
23        # Zero the gradient
24        img.grad.data.zero_()
25
26        # Clip the image values to be in the valid range
27        img.data = torch.clamp(img.data, 0, 1)
28
29    # Save the final deep dream image
30    result = transforms.ToPILImage()(img.squeeze(0).cpu())
31    result.save(output_path)
```

Svrha: Ova funkcija generiše DeepDream sliku iterativnim poboljšanjem obrazaca u ulaznoj slici da bi se maksimizovale aktivacije u odredjenom sloju neuronske mreže.

Parametri:

- `image_tensor`: Ulazna slika kao PyTorch tensor.
- `model`: Osnovni model neuronske mreže (npr. InceptionV3).
- `layer_idx`: Indeks sloja u modelu čije aktivacije želimo da maksimizujemo.
- `iterations`: Broj iteracija za izvodjenje procesa DeepDream.
- `lr`: Stopa učenja za optimizator.
- `octave_scale`: Faktor razmere za ažuriranje tensora slike.
- `output_path`: Putanja za čuvanje rezultujuće DeepDream slike.

7.1 Inicijalno podešavanje

```
1 # Convert image tensor to nn.Parameter
2 img = nn.Parameter(image_tensor.to(device))
3
4 # Define the deep dream model
5 dream_model = DeepDream(model, layer_idx).to(device)
6
7 # Define the optimizer
8 optimizer = optim.Adam([img], lr=lr)
```

- **Konvertuj tensor slike:** Konvertuje tensor ulazne slike u `nn.Parameter`, koji omogućava da se gradijenti izračunaju u odnosu na sliku.
- **Definisati DeepDream model:** Inicijalizuje DeepDream model sa datim osnovnim modelom i indeksom sloja i premešta ga na odgovarajući uređaj (npr. GPU ili CPU).
- **Definišite optimizator:** Podešava Adam optimizator da optimizuje parametar `img` sa navedenom stopom učenja.

7.2 DeepDream iteracije

```
1 for i in range(iterations):
2     optimizer.zero_grad()
3     features = dream_model(img)
4     loss = features.norm()
5     loss.backward()
6     optimizer.step()
7
8     # Apply the octave scaling
9     img.data = img.data + octave_scale * img.grad.data
10
11     # Zero the gradient
12     img.grad.data.zero_()
13
14     # Clip the image values to be in the valid range
15     img.data = torch.clamp(img.data, 0, 1)
```

- **Nula gradijent:** Briše stare gradijente iz optimizatora.
- **Forward prolaz:** Propušta sliku kroz DeepDream model da bi dobio aktivacije (funkcije) sa navedenog sloja.
- **Compute Loss:** Izračunava gubitak kao normu karakteristika. Norma predstavlja veličinu aktivacija.
- **Backward prolaz:** Izračunava gradijente u odnosu na gubitak.
- **Korak optimizatora:** Ažurira sliku na osnovu izračunatih gradijenata.

- **Primena oktavnog skaliranja:** Modifikuje podatke slike koristeći izračunate gradijente i navedenu oktavnu skalu.
- **Postaviti gradijent na nulu:** Briše gradijente slike.
- **Saseći vrednosti slike:** Obezbedjuje da vrednosti podataka slike ostanu u važećem opsegu $[0, 1]$.

7.3 Čuvanje rezultata

```

1 # Save the final deep dream image
2 result = transforms.ToPILImage()(img.squeeze(0).cpu())
3 result.save(output_path)

```

- **Pretvoriti u PIL sliku:** Konvertuje konačni tensor slike u PIL sliku nakon stiskanja i premeštanja u CPU.
- **Sačuvati sliku:** Čuva PIL sliku na navedenoj izlaznoj putanji.

7.4 Rezime

Funkcija `deep_dream` obavlja sledeće korake:

- 1. **Inicijalno podešavanje:** Priprema tensor slike, model i optimizator.
- 2. **Iterativni proces:** Poboljšava obrasce na slici maksimizovanjem aktivacija u određenom sloju modela:
 - Izračunava prolaz unapred da bi dobio karakteristike.
 - Izračunava i širi gubitak unazad.
 - Ažurira sliku radi poboljšanja obrazaca.
 - Primenjuje oktavno skaliranje i iseče vrednosti slike.
- 3. **Čuvanje rezultata:** Konvertuje konačni tensor u sliku i čuva je.

7.5 Upotreba

Funkcija se obično koristi za generisanje vizuelno privlačnih slika nalik snu korišćenjem mapa karakteristika koje su naučile duboke konvolutivne mreže. Ističe i pojačava obrasce koje prepoznaje model, što rezultira nadrealnim i apstraktnim slikama.

8 Podešavanje parametara

```
1 # Set the device
2 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
3
4 # Specify the layer index for deep dream (you can experiment with
   different layers)
5 layer_index = 10
6
7 iterations = 20
8 learning_rate = 0.01
9 octave_scale = 1.4
```

9 Podešavanje težina klasa

```
1 class_weights = [0.42, 0.58]
2 class_weights = torch.tensor(class_weights).to(device)
```

10 Treniranje i obučavanje modela

```
1 loss_fn = nn.CrossEntropyLoss(weight=class_weights)
2 optimizer = optim.Adam(model.parameters())
3 model.to(device)
4
5 num_epochs = 10
6 best_accuracy = 0.0
7 best_labels = None
8 best_preds = None
9
10 for epoch in range(num_epochs):
11     print(f"Epoch {epoch+1}/{num_epochs}")
12     train_loop(train_dataloader, model, loss_fn, optimizer, device)
13     accuracy, all_labels, all_preds = test_loop(test_dataloader, model,
        loss_fn, device)
14
15     if accuracy > best_accuracy:
16         best_accuracy = accuracy
17         best_labels = all_labels
18         best_preds = all_preds
19
20 print(f'Best Accuracy: {best_accuracy}')
```

11 Primena modela na skup podataka

U ovom projektu sam koristio skup podataka u kome se nalaze slike pasa i mačaka. U trening skupu se nalazi 2000 slika (1000 slika mačaka i 1000 slika pasa) i još 1000 slika u test skupu (500 slika mačaka i 500 slika pasa). Cilj mreže

je da prepozna da li se na slici nalazi pas ili mačka. Koristio sam svoju mrežu, kao i pretreniranu i uporedio rezultate.

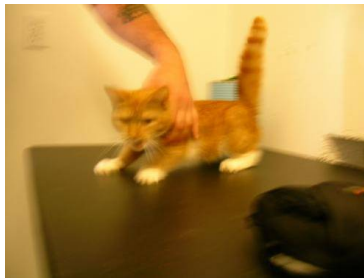


Figure 2: Regular cat

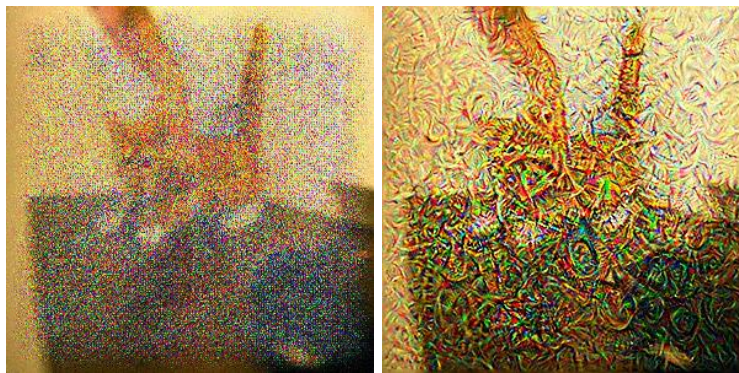


Figure 3: Manual vs Pretrained CNN

12 Rezultati

Nakon 10 epoha dobijaju se sledeći rezultati:

- Za moju neuronsku mrežu:
 - tačnost: 0.663
 - matrica konfuzije:
 -
- Za pretreniranu neuronsku mrežu:
 - tačnost: 0.736
 - matrica konfuzije:

Na osnovu mere tačnosti i matrice konfuzije možemo zaključiti da pretrenirana neuronska mreža bolje klasifikuje slike od moje neuronske mreže.

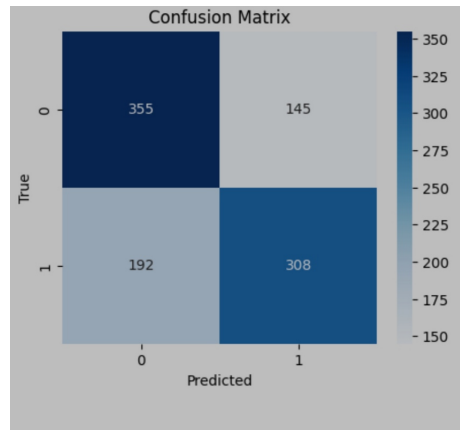


Figure 4: Conf matrix for my CNN

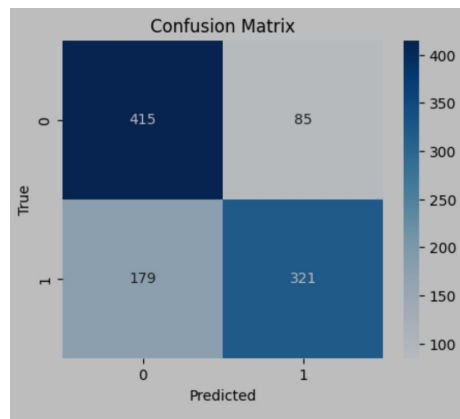


Figure 5: Conf matrix for pretrained CNN

13 Literatura

- A Systematic Review of Deep Dream
- Deep Dream using Tensorflow
- PyTorch Documentation