In [ ]:
```
!pip install transformers
!pip install -U pip setuptools wheel
!pip install -U spacy
!python -m spacy download en_core_web_sm
!pip install bert-extractive-summarizer
```

```
Collecting transformers
  Downloading https://files.pythonhosted.org/packages/f9/54/5ca07ec9569
d2f232f3166de5457b63943882f7950ddfcc887732fc7fb23/transformers-4.3.3-py
3-none-any.whl (1.9MB)
     |████████████████████████████████| 1.9MB 17.2MB/s
Requirement already satisfied: packaging in /usr/local/lib/python3.7/di
st-packages (from transformers) (20.9)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/
dist-packages (from transformers) (1.19.5)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dis
t-packages (from transformers) (2.23.0)
Requirement already satisfied: importlib-metadata; python_version < "3.
8" in /usr/local/lib/python3.7/dist-packages (from transformers) (3.7.
0)
Collecting tokenizers<0.11,>=0.10.1
  Downloading https://files.pythonhosted.org/packages/71/23/2ddc317b212
1117bf34dd00f5b0de194158f2a44ee2bf5e47c7166878a97/tokenizers-0.10.1-cp3
7-cp37m-manylinux2010_x86_64.whl (3.2MB)
     |████████████████████████████████| 3.2MB 50.5MB/s
Collecting sacremoses
  Downloading https://files.pythonhosted.org/packages/7d/34/09d19aff26e
dcc8eb2a01bed8e98f13a1537005d31e95233fd48216eed10/sacremoses-0.0.43.ta
r.gz (883kB)
     |████████████████████████████████| 890kB 44.4MB/s
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/d
ist-packages (from transformers) (4.41.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/pyth
on3.7/dist-packages (from transformers) (2019.12.20)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dis
t-packages (from transformers) (3.0.12)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/pytho
n3.7/dist-packages (from packaging->transformers) (2.4.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/pyt
hon3.7/dist-packages (from requests->transformers) (2020.12.5)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.
7/dist-packages (from requests->transformers) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/pyth
on3.7/dist-packages (from requests->transformers) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
in /usr/local/lib/python3.7/dist-packages (from requests->transformers)
(1.24.3)
Requirement already satisfied: typing-extensions>=3.6.4; python_version
< "3.8" in /usr/local/lib/python3.7/dist-packages (from importlib-metad
ata; python_version < "3.8"->transformers) (3.7.4.3)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/di
st-packages (from importlib-metadata; python_version < "3.8"->transform
ers) (3.4.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-pac
kages (from sacremoses->transformers) (1.15.0)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-p
ackages (from sacremoses->transformers) (7.1.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-
packages (from sacremoses->transformers) (1.0.1)
Building wheels for collected packages: sacremoses
  Building wheel for sacremoses (setup.py) ... done
  Created wheel for sacremoses: filename=sacremoses-0.0.43-cp37-none-an
y.whl size=893262 sha256=178e5d05a9e5d40369539b3d2fbe4fc4df778cb272388a
27206262fc903d0cbb
  Stored in directory: /root/.cache/pip/wheels/29/3c/fd/7ce5c3f0666dab3
1a50123635e6fb5e19ceb42ce38d4e58f45
Successfully built sacremoses
Installing collected packages: tokenizers, sacremoses, transformers
Successfully installed sacremoses-0.0.43 tokenizers-0.10.1 transformers
-4.3.3
Collecting pip
  Downloading https://files.pythonhosted.org/packages/fe/ef/60d7ba03b5c
442309ef42e7d69959f73aacccd0d86008362a681c4698e83/pip-21.0.1-py3-none-a
ny.whl (1.5MB)
     |████████████████████████████████| 1.5MB 17.1MB/s
Collecting setuptools
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
  Downloading https://files.pythonhosted.org/packages/4e/ca/0456249eead
852a957d306feb0cce454b823cbc3bea7821971febfa45f99/setuptools-53.1.0-py3
-none-any.whl (784kB)
     |████████████████████████████████| 788kB 55.6MB/s
Requirement already up-to-date: wheel in /usr/local/lib/python3.7/dist-
packages (0.36.2)
ERROR: datascience 0.10.6 has requirement folium==0.2.1, but you'll hav
e folium 0.8.3 which is incompatible.
Installing collected packages: pip, setuptools
  Found existing installation: pip 19.3.1
    Uninstalling pip-19.3.1:
      Successfully uninstalled pip-19.3.1
  Found existing installation: setuptools 53.0.0
    Uninstalling setuptools-53.0.0:
      Successfully uninstalled setuptools-53.0.0
Successfully installed pip-21.0.1 setuptools-53.1.0

Requirement already satisfied: spacy in /usr/local/lib/python3.7/dist-p
ackages (2.2.4)
Collecting spacy
  Downloading spacy-3.0.3-cp37-cp37m-manylinux2014_x86_64.whl (12.7 MB)
     |████████████████████████████████| 12.7 MB 206 kB/s
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-
packages (from spacy) (2.11.3)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/py
thon3.7/dist-packages (from spacy) (2.0.5)
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.
7/dist-packages (from spacy) (1.19.5)
Requirement already satisfied: wasabi<1.1.0,>=0.8.1 in /usr/local/lib/p
ython3.7/dist-packages (from spacy) (0.8.2)
Collecting catalogue<2.1.0,>=2.0.1
  Downloading catalogue-2.0.1-py3-none-any.whl (9.6 kB)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python
3.7/dist-packages (from spacy) (20.9)
Collecting thinc<8.1.0,>=8.0.0
  Downloading thinc-8.0.1-cp37-cp37m-manylinux2014_x86_64.whl (1.1 MB)
     |████████████████████████████████| 1.1 MB 66.5 MB/s
Requirement already satisfied: typing-extensions>=3.7.4 in /usr/local/l
ib/python3.7/dist-packages (from spacy) (3.7.4.3)
Collecting typer<0.4.0,>=0.3.0
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
  Downloading typer-0.3.2-py3-none-any.whl (21 kB)
Requirement already satisfied: blis<0.8.0,>=0.4.0 in /usr/local/lib/pyt
hon3.7/dist-packages (from spacy) (0.4.1)
Collecting pathy
  Downloading pathy-0.4.0-py3-none-any.whl (36 kB)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/d
ist-packages (from spacy) (53.1.0)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/
lib/python3.7/dist-packages (from spacy) (1.0.5)
Requirement already satisfied: importlib-metadata>=0.20 in /usr/local/l
ib/python3.7/dist-packages (from spacy) (3.7.0)
Collecting srsly<3.0.0,>=2.4.0
  Downloading srsly-2.4.0-cp37-cp37m-manylinux2014_x86_64.whl (456 kB)
     |████████████████████████████████| 456 kB 51.0 MB/s
Collecting spacy-legacy<3.1.0,>=3.0.0
  Downloading spacy_legacy-3.0.1-py2.py3-none-any.whl (7.0 kB)
Collecting pydantic<1.8.0,>=1.7.1
  Downloading pydantic-1.7.3-cp37-cp37m-manylinux2014_x86_64.whl (9.1 M
B)
     |████████████████████████████████| 9.1 MB 66.9 MB/s
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/py
thon3.7/dist-packages (from spacy) (4.41.1)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/li
b/python3.7/dist-packages (from spacy) (2.23.0)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/
python3.7/dist-packages (from spacy) (3.0.5)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/di
st-packages (from importlib-metadata>=0.20->spacy) (3.4.0)
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/pytho
n3.7/dist-packages (from packaging>=20.0->spacy) (2.4.7)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.
7/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/pyth
on3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/pyt
hon3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2020.12.5)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
in /usr/local/lib/python3.7/dist-packages (from requests<3.0.0,>=2.13.0
->spacy) (1.24.3)
Requirement already satisfied: click<7.2.0,>=7.1.1 in /usr/local/lib/py
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
thon3.7/dist-packages (from typer<0.4.0,>=0.3.0->spacy) (7.1.2)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/pytho
n3.7/dist-packages (from jinja2->spacy) (1.1.1)
Collecting smart-open<4.0.0,>=2.2.0
  Downloading smart_open-3.0.0.tar.gz (113 kB)
     |████████████████████████████████| 113 kB 77.9 MB/s
Building wheels for collected packages: smart-open
  Building wheel for smart-open (setup.py) ... done
  Created wheel for smart-open: filename=smart_open-3.0.0-py3-none-any.
whl size=107097 sha256=b2f99ec900abdda1faf9f17d7329411071bd6c9c99a87302
91bea2b9cc567285
  Stored in directory: /root/.cache/pip/wheels/83/a6/12/bf3c1a667bde425
1be5b7a3368b2d604c9af2105b5c1cb1870
Successfully built smart-open
Installing collected packages: catalogue, typer, srsly, smart-open, pyd
antic, thinc, spacy-legacy, pathy, spacy
  Attempting uninstall: catalogue
    Found existing installation: catalogue 1.0.0
    Uninstalling catalogue-1.0.0:
      Successfully uninstalled catalogue-1.0.0
  Attempting uninstall: srsly
    Found existing installation: srsly 1.0.5
    Uninstalling srsly-1.0.5:
      Successfully uninstalled srsly-1.0.5
  Attempting uninstall: smart-open
    Found existing installation: smart-open 4.2.0
    Uninstalling smart-open-4.2.0:
      Successfully uninstalled smart-open-4.2.0
  Attempting uninstall: thinc
    Found existing installation: thinc 7.4.0
    Uninstalling thinc-7.4.0:
      Successfully uninstalled thinc-7.4.0
  Attempting uninstall: spacy
    Found existing installation: spacy 2.2.4
    Uninstalling spacy-2.2.4:
      Successfully uninstalled spacy-2.2.4
Successfully installed catalogue-2.0.1 pathy-0.4.0 pydantic-1.7.3 smart
-open-3.0.0 spacy-3.0.3 spacy-legacy-3.0.1 srsly-2.4.0 thinc-8.0.1 type
r-0.3.2
2021-02-28 17:47:07.876376: I tensorflow/stream_executor/platform/defau
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
lt/dso_loader.cc:49] Successfully opened dynamic library libcudart.so.1
0.1
Collecting en-core-web-sm==3.0.0
  Downloading https://github.com/explosion/spacy-models/releases/downlo
ad/en_core_web_sm-3.0.0/en_core_web_sm-3.0.0-py3-none-any.whl (13.7 MB)
     |████████████████████████████████| 13.7 MB 79 kB/s
Requirement already satisfied: spacy<3.1.0,>=3.0.0 in /usr/local/lib/py
thon3.7/dist-packages (from en-core-web-sm==3.0.0) (3.0.3)
Requirement already satisfied: thinc<8.1.0,>=8.0.0 in /usr/local/lib/py
thon3.7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.0.0)
(8.0.1)
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.
7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.0.0) (1.1
9.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-
packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.0.0) (2.11.3)
Requirement already satisfied: typer<0.4.0,>=0.3.0 in /usr/local/lib/py
thon3.7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.0.0)
(0.3.2)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/py
thon3.7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.0.0)
(4.41.1)
Requirement already satisfied: catalogue<2.1.0,>=2.0.1 in /usr/local/li
b/python3.7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.
0.0) (2.0.1)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/
lib/python3.7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==
3.0.0) (1.0.5)
Requirement already satisfied: typing-extensions>=3.7.4 in /usr/local/l
ib/python3.7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==
3.0.0) (3.7.4.3)
Requirement already satisfied: blis<0.8.0,>=0.4.0 in /usr/local/lib/pyt
hon3.7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.0.0)
(0.4.1)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python
3.7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.0.0) (2
0.9)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/d
ist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.0.0) (53.1.0)
Requirement already satisfied: srsly<3.0.0,>=2.4.0 in /usr/local/lib/py
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
thon3.7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.0.0)
(2.4.0)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/
python3.7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.0.
0) (3.0.5)
Requirement already satisfied: importlib-metadata>=0.20 in /usr/local/l
ib/python3.7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==
3.0.0) (3.7.0)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/py
thon3.7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.0.0)
(2.0.5)
Requirement already satisfied: pathy in /usr/local/lib/python3.7/dist-p
ackages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.0.0) (0.4.0)
Requirement already satisfied: wasabi<1.1.0,>=0.8.1 in /usr/local/lib/p
ython3.7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.0.
0) (0.8.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/li
b/python3.7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.
0.0) (2.23.0)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.0 in /usr/loca
l/lib/python3.7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm
==3.0.0) (3.0.1)
Requirement already satisfied: pydantic<1.8.0,>=1.7.1 in /usr/local/li
b/python3.7/dist-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.
0.0) (1.7.3)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/di
st-packages (from importlib-metadata>=0.20->spacy<3.1.0,>=3.0.0->en-cor
e-web-sm==3.0.0) (3.4.0)
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/pytho
n3.7/dist-packages (from packaging>=20.0->spacy<3.1.0,>=3.0.0->en-core-
web-sm==3.0.0) (2.4.7)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
in /usr/local/lib/python3.7/dist-packages (from requests<3.0.0,>=2.13.0
->spacy<3.1.0,>=3.0.0->en-core-web-sm==3.0.0) (1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/pyth
on3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy<3.1.0,>=3.0.0-
>en-core-web-sm==3.0.0) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.
7/dist-packages (from requests<3.0.0,>=2.13.0->spacy<3.1.0,>=3.0.0->en-
core-web-sm==3.0.0) (2.10)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/pyt
hon3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy<3.1.0,>=3.0.0
->en-core-web-sm==3.0.0) (2020.12.5)
Requirement already satisfied: click<7.2.0,>=7.1.1 in /usr/local/lib/py
thon3.7/dist-packages (from typer<0.4.0,>=0.3.0->spacy<3.1.0,>=3.0.0->e
n-core-web-sm==3.0.0) (7.1.2)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/pytho
n3.7/dist-packages (from jinja2->spacy<3.1.0,>=3.0.0->en-core-web-sm==
3.0.0) (1.1.1)
Requirement already satisfied: smart-open<4.0.0,>=2.2.0 in /usr/local/l
ib/python3.7/dist-packages (from pathy->spacy<3.1.0,>=3.0.0->en-core-we
b-sm==3.0.0) (3.0.0)
Installing collected packages: en-core-web-sm
  Attempting uninstall: en-core-web-sm
    Found existing installation: en-core-web-sm 2.2.5
    Uninstalling en-core-web-sm-2.2.5:
      Successfully uninstalled en-core-web-sm-2.2.5
Successfully installed en-core-web-sm-3.0.0
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
Collecting bert-extractive-summarizer
  Downloading bert_extractive_summarizer-0.7.0-py3-none-any.whl (17 kB)
Requirement already satisfied: spacy in /usr/local/lib/python3.7/dist-p
ackages (from bert-extractive-summarizer) (3.0.3)
Requirement already satisfied: transformers in /usr/local/lib/python3.
7/dist-packages (from bert-extractive-summarizer) (4.3.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.
7/dist-packages (from bert-extractive-summarizer) (0.22.2.post1)
Requirement already satisfied: scipy>=0.17.0 in /usr/local/lib/python3.
7/dist-packages (from scikit-learn->bert-extractive-summarizer) (1.4.1)
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.
7/dist-packages (from scikit-learn->bert-extractive-summarizer) (1.19.
5)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.
7/dist-packages (from scikit-learn->bert-extractive-summarizer) (1.0.1)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/
lib/python3.7/dist-packages (from spacy->bert-extractive-summarizer)
 (1.0.5)
Requirement already satisfied: blis<0.8.0,>=0.4.0 in /usr/local/lib/pyt
hon3.7/dist-packages (from spacy->bert-extractive-summarizer) (0.4.1)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.0 in /usr/loca
l/lib/python3.7/dist-packages (from spacy->bert-extractive-summarizer)
 (3.0.1)
Requirement already satisfied: catalogue<2.1.0,>=2.0.1 in /usr/local/li
b/python3.7/dist-packages (from spacy->bert-extractive-summarizer) (2.
0.1)
Requirement already satisfied: importlib-metadata>=0.20 in /usr/local/l
ib/python3.7/dist-packages (from spacy->bert-extractive-summarizer) (3.
7.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python
3.7/dist-packages (from spacy->bert-extractive-summarizer) (20.9)
Requirement already satisfied: wasabi<1.1.0,>=0.8.1 in /usr/local/lib/p
ython3.7/dist-packages (from spacy->bert-extractive-summarizer) (0.8.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-
packages (from spacy->bert-extractive-summarizer) (2.11.3)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/py
thon3.7/dist-packages (from spacy->bert-extractive-summarizer) (2.0.5)
Requirement already satisfied: typer<0.4.0,>=0.3.0 in /usr/local/lib/py
thon3.7/dist-packages (from spacy->bert-extractive-summarizer) (0.3.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/li
b/python3.7/dist-packages (from spacy->bert-extractive-summarizer) (2.2
3.0)
Requirement already satisfied: pathy in /usr/local/lib/python3.7/dist-p
ackages (from spacy->bert-extractive-summarizer) (0.4.0)
Requirement already satisfied: srsly<3.0.0,>=2.4.0 in /usr/local/lib/py
thon3.7/dist-packages (from spacy->bert-extractive-summarizer) (2.4.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/py
thon3.7/dist-packages (from spacy->bert-extractive-summarizer) (4.41.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/d
ist-packages (from spacy->bert-extractive-summarizer) (53.1.0)
Requirement already satisfied: typing-extensions>=3.7.4 in /usr/local/l
ib/python3.7/dist-packages (from spacy->bert-extractive-summarizer) (3.
7.4.3)
Requirement already satisfied: pydantic<1.8.0,>=1.7.1 in /usr/local/li
b/python3.7/dist-packages (from spacy->bert-extractive-summarizer) (1.
7.3)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/
python3.7/dist-packages (from spacy->bert-extractive-summarizer) (3.0.
5)
Requirement already satisfied: thinc<8.1.0,>=8.0.0 in /usr/local/lib/py
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
thon3.7/dist-packages (from spacy->bert-extractive-summarizer) (8.0.1)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/di
st-packages (from importlib-metadata>=0.20->spacy->bert-extractive-summ
arizer) (3.4.0)
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/pytho
n3.7/dist-packages (from packaging>=20.0->spacy->bert-extractive-summar
izer) (2.4.7)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
 in /usr/local/lib/python3.7/dist-packages (from requests<3.0.0,>=2.13.
0->spacy->bert-extractive-summarizer) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.
7/dist-packages (from requests<3.0.0,>=2.13.0->spacy->bert-extractive-s
ummarizer) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/pyt
hon3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy->bert-extract
ive-summarizer) (2020.12.5)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/pyth
on3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy->bert-extracti
ve-summarizer) (3.0.4)
Requirement already satisfied: click<7.2.0,>=7.1.1 in /usr/local/lib/py
thon3.7/dist-packages (from typer<0.4.0,>=0.3.0->spacy->bert-extractive
-summarizer) (7.1.2)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/pytho
n3.7/dist-packages (from jinja2->spacy->bert-extractive-summarizer) (1.
1.1)
Requirement already satisfied: smart-open<4.0.0,>=2.2.0 in /usr/local/l
ib/python3.7/dist-packages (from pathy->spacy->bert-extractive-summariz
er) (3.0.0)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/pyth
on3.7/dist-packages (from transformers->bert-extractive-summarizer) (20
19.12.20)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dis
t-packages (from transformers->bert-extractive-summarizer) (3.0.12)
Requirement already satisfied: sacremoses in /usr/local/lib/python3.7/d
ist-packages (from transformers->bert-extractive-summarizer) (0.0.43)
Requirement already satisfied: tokenizers<0.11,>=0.10.1 in /usr/local/l
ib/python3.7/dist-packages (from transformers->bert-extractive-summariz
er) (0.10.1)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-pac
kages (from sacremoses->transformers->bert-extractive-summarizer) (1.1
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
5.0)
Installing collected packages: bert-extractive-summarizer
Successfully installed bert-extractive-summarizer-0.7.0
```

In [ ]:
```python
import spacy, random, re, nltk
nltk.download('punkt')
nltk.download('stopwords')
from nltk.tokenize import word_tokenize
from transformers import pipeline, set_seed
from summarizer import Summarizer
from nltk.corpus import stopwords
from nltk.collocations import *

nlp = spacy.load("en_core_web_sm")
model = Summarizer()
generator = pipeline('text-generation', model='gpt2')
set_seed(42)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Some weights of GPT2Model were not initialized from the model checkpoin
t at gpt2 and are newly initialized: ['h.0.attn.masked_bias', 'h.1.att
n.masked_bias', 'h.2.attn.masked_bias', 'h.3.attn.masked_bias', 'h.4.at
tn.masked_bias', 'h.5.attn.masked_bias', 'h.6.attn.masked_bias', 'h.7.a
ttn.masked_bias', 'h.8.attn.masked_bias', 'h.9.attn.masked_bias', 'h.1
0.attn.masked_bias', 'h.11.attn.masked_bias']
You should probably TRAIN this model on a down-stream task to be able t
o use it for predictions and inference.
```

In [ ]:
```
input_to_model = "Today is snowy day at"
text = generator(input_to_model, max_length=100, num_return_sequences=1
)[0]['generated_text'].replace('\n', '').replace('\'', '')
```

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
```

In [ ]:
```
doc = nlp(text)
assert doc.has_annotation("SENT_START")
s = [sent.text.capitalize() for sent in doc.sents]
s = " ".join(s[:len(s)-2])
s
```

Out[ ]:
```
'Today is snowy day at the intersection of south and west bitter street
s. I stopped at a gas station to buy the next 3 gallons of warm water t
o drink on the way down. Here are the locations to go if possible. Sout
h1.'
```

In [ ]:
```
sentence = model(s)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
sentence
```

Out[ ]:
```
'Today is snowy day at the intersection of south and west bitter street
s.'
```

In [ ]:
```
words = []
for token in nlp(sentence):
  if token.pos_ in ['NOUN', 'ADJ']:
    words.append(token.orth_)
s=" ".join(words)
s
```

Out[ ]:
```
'Today snowy day intersection south west bitter streets'
```

Description

This notebook utilizes CLIP to steer BigGAN towards images that match a textual prompt.

It also dings.

Restart the kernel...

and rerun all cells after running the cell below and setting your prompt.

In the top menu press \<Runtime> then \<Restart and run all> to restart everything after this cell stops running

In [ ]:
```
import subprocess

CUDA_version = [s for s in subprocess.check_output(["nvcc", "--version"
]).decode("UTF-8").split(", ") if s.startswith("release")][0].split(" "
)[-1]
print("CUDA version:", CUDA_version)

if CUDA_version == "10.0":
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    torch_version_suffix = "+cu100"
elif CUDA_version == "10.1":
    torch_version_suffix = "+cu101"
elif CUDA_version == "10.2":
    torch_version_suffix = ""
else:
    torch_version_suffix = "+cu110"

!pip install torch==1.7.1{torch_version_suffix} torchvision==0.8.2{torc
h_version_suffix} -f https://download.pytorch.org/whl/torch_stable.html
ftfy regex
```

```
CUDA version: 10.1
Looking in links: https://download.pytorch.org/whl/torch_stable.html
Requirement already satisfied: torch==1.7.1+cu101 in /usr/local/lib/pyt
hon3.7/dist-packages (1.7.1+cu101)
Requirement already satisfied: torchvision==0.8.2+cu101 in /usr/local/l
ib/python3.7/dist-packages (0.8.2+cu101)
Collecting ftfy
  Downloading ftfy-5.9.tar.gz (66 kB)
     |████████████████████████████████| 66 kB 4.7 MB/s
Requirement already satisfied: regex in /usr/local/lib/python3.7/dist-p
ackages (2019.12.20)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-p
ackages (from torch==1.7.1+cu101) (1.19.5)
Requirement already satisfied: typing-extensions in /usr/local/lib/pyth
on3.7/dist-packages (from torch==1.7.1+cu101) (3.7.4.3)
Requirement already satisfied: pillow>=4.1.1 in /usr/local/lib/python3.
7/dist-packages (from torchvision==0.8.2+cu101) (7.0.0)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.7/dist
-packages (from ftfy) (0.2.5)
Building wheels for collected packages: ftfy
  Building wheel for ftfy (setup.py) ... done
  Created wheel for ftfy: filename=ftfy-5.9-py3-none-any.whl size=46451
sha256=f35ba15ac1be0db9cd254f9059863e8746427e01d6e514d9a39cea803ee77f7b
  Stored in directory: /root/.cache/pip/wheels/4f/6b/a5/84880e943570765
9c6b96d3aadeb9a87a41f61ec9ede469f41
Successfully built ftfy
Installing collected packages: ftfy
Successfully installed ftfy-5.9
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Many Imports

Import the usual libraries

In [ ]:
```
import torch
import numpy as np
import torchvision
import torchvision.transforms.functional as TF

import PIL
import matplotlib.pyplot as plt

import os
import random
import imageio
from IPython import display
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

import glob

from google.colab import output
```

In [ ]:
```
!nvidia-smi -L
```

```
GPU 0: Tesla T4 (UUID: GPU-04ffe672-9fb2-7a6e-fdd1-1809e4637880)
```

Perceptor

Get and load up CLIP

In [ ]:
```
%cd /content/

!git clone https://github.com/openai/CLIP.git
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
%cd /content/CLIP/

!pip install ftfy

import os
import clip
import torch


import numpy as np

# Load the model
perceptor, preprocess = clip.load('ViT-B/32')
```

```
/content
Cloning into 'CLIP'...
remote: Enumerating objects: 64, done.
remote: Total 64 (delta 0), reused 0 (delta 0), pack-reused 64
Unpacking objects: 100% (64/64), done.
/content/CLIP
Requirement already satisfied: ftfy in /usr/local/lib/python3.7/dist-pa
ckages (5.9)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.7/dist
-packages (from ftfy) (0.2.5)
```

```
100%|████████████████████████████████████████| 354M/354M [00:02<00:00,
125MiB/s]
```

Parameters

Don't bother changing anything but the prompt below if you're not using a different type of BigGAN

In [ ]:
```
im_shape = [512, 512, 3]
sideX, sideY, channels = im_shape
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
tx = clip.tokenize(s)
```

Define

Some helper functions

In [ ]:

```
def displ(img, pre_scaled=True):
  img = np.array(img)[:,:,:]
  img = np.transpose(img, (1, 2, 0))
  if not pre_scaled:
    img = scale(img, 48*4, 32*4)
  imageio.imwrite(str(3) + '.png', np.array(img))
  return display.Image(str(3)+'.png')

def gallery(array, ncols=2):
    nindex, height, width, intensity = array.shape
    nrows = nindex//ncols
    assert nindex == nrows*ncols
    # want result.shape = (height*nrows, width*ncols, intensity)
    result = (array.reshape(nrows, ncols, height, width, intensity)
              .swapaxes(1,2)
              .reshape(height*nrows, width*ncols, intensity))
    return result

def card_padded(im, to_pad=3):
  return np.pad(np.pad(np.pad(im, [[1,1], [1,1], [0,0]],constant_values
=0), [[2,2], [2,2], [0,0]],constant_values=1),
            [[to_pad,to_pad], [to_pad,to_pad], [0,0]],constant_values=0
)

def get_all(img):
  img = np.transpose(img, (0,2,3,1))
  cards = np.zeros((img.shape[0], sideX+12, sideY+12, 3))
  for i in range(len(img)):
    cards[i] = card_padded(img[i])
  print(img.shape)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
  cards = gallery(cards)
  imageio.imwrite(str(3) + '.png', np.array(cards))
  return display.Image(str(3)+'.png')
```

Generator

Alter BigGAN for multiple class & latent inputs

In [ ]:

```
!pip install pytorch-pretrained-biggan

import torch.nn as nn
import torch.nn.functional as F
import math
import json
import logging
import os
import shutil
import tempfile
from functools import wraps
from hashlib import sha256
import sys
from io import open

import boto3
import requests
from botocore.exceptions import ClientError
from tqdm import tqdm

try:
    from urllib.parse import urlparse
except ImportError:
    from urlparse import urlparse

try:
    from pathlib import Path
    PYTORCH_PRETRAINED_BIGGAN_CACHE = Path(os.getenv('PYTORCH_PRETRAINE
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
D_BIGGAN_CACHE',
                                                Path.home() / '.pyto
rch_pretrained_biggan'))
except (AttributeError, ImportError):
    PYTORCH_PRETRAINED_BIGGAN_CACHE = os.getenv('PYTORCH_PRETRAINED_BIG
GAN_CACHE',
                                                os.path.join(os.path.expa
nduser("~"), '.pytorch_pretrained_biggan'))

logger = logging.getLogger(__name__)  # pylint: disable=invalid-name


def url_to_filename(url, etag=None):
    """
    Convert `url` into a hashed filename in a repeatable way.
    If `etag` is specified, append its hash to the url's, delimited
    by a period.
    """
    url_bytes = url.encode('utf-8')
    url_hash = sha256(url_bytes)
    filename = url_hash.hexdigest()

    if etag:
        etag_bytes = etag.encode('utf-8')
        etag_hash = sha256(etag_bytes)
        filename += '.' + etag_hash.hexdigest()

    return filename


def filename_to_url(filename, cache_dir=None):
    """
    Return the url and etag (which may be ``None``) stored for `filenam
e`.
    Raise ``EnvironmentError`` if `filename` or its stored metadata do
 not exist.
    """
    if cache_dir is None:
        cache_dir = PYTORCH_PRETRAINED_BIGGAN_CACHE
    if sys.version_info[0] == 3 and isinstance(cache_dir, Path):
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        cache_dir = str(cache_dir)

    cache_path = os.path.join(cache_dir, filename)
    if not os.path.exists(cache_path):
        raise EnvironmentError("file {} not found".format(cache_path))

    meta_path = cache_path + '.json'
    if not os.path.exists(meta_path):
        raise EnvironmentError("file {} not found".format(meta_path))

    with open(meta_path, encoding="utf-8") as meta_file:
        metadata = json.load(meta_file)
    url = metadata['url']
    etag = metadata['etag']

    return url, etag


def cached_path(url_or_filename, cache_dir=None):
    """
    Given something that might be a URL (or might be a local path),
    determine which. If it's a URL, download the file and cache it, and
    return the path to the cached file. If it's already a local path,
    make sure the file exists and then return the path.
    """
    if cache_dir is None:
        cache_dir = PYTORCH_PRETRAINED_BIGGAN_CACHE
    if sys.version_info[0] == 3 and isinstance(url_or_filename, Path):
        url_or_filename = str(url_or_filename)
    if sys.version_info[0] == 3 and isinstance(cache_dir, Path):
        cache_dir = str(cache_dir)

    parsed = urlparse(url_or_filename)

    if parsed.scheme in ('http', 'https', 's3'):
        # URL, so get it from the cache (downloading if necessary)
        return get_from_cache(url_or_filename, cache_dir)
    elif os.path.exists(url_or_filename):
        # File, and it exists.
        return url_or_filename
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    elif parsed.scheme == '':
        # File, but it doesn't exist.
        raise EnvironmentError("file {} not found".format(url_or_filena
me))
    else:
        # Something unknown
        raise ValueError("unable to parse {} as a URL or as a local pat
h".format(url_or_filename))


def split_s3_path(url):
    """Split a full s3 path into the bucket name and path."""
    parsed = urlparse(url)
    if not parsed.netloc or not parsed.path:
        raise ValueError("bad s3 path {}".format(url))
    bucket_name = parsed.netloc
    s3_path = parsed.path
    # Remove '/' at beginning of path.
    if s3_path.startswith("/"):
        s3_path = s3_path[1:]
    return bucket_name, s3_path


def s3_request(func):
    """
    Wrapper function for s3 requests in order to create more helpful er
ror
    messages.
    """

    @wraps(func)
    def wrapper(url, *args, **kwargs):
        try:
            return func(url, *args, **kwargs)
        except ClientError as exc:
            if int(exc.response["Error"]["Code"]) == 404:
                raise EnvironmentError("file {} not found".format(url))
            else:
                raise
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    return wrapper


@s3_request
def s3_etag(url):
    """Check ETag on S3 object."""
    s3_resource = boto3.resource("s3")
    bucket_name, s3_path = split_s3_path(url)
    s3_object = s3_resource.Object(bucket_name, s3_path)
    return s3_object.e_tag


@s3_request
def s3_get(url, temp_file):
    """Pull a file directly from S3."""
    s3_resource = boto3.resource("s3")
    bucket_name, s3_path = split_s3_path(url)
    s3_resource.Bucket(bucket_name).download_fileobj(s3_path, temp_file
)


def http_get(url, temp_file):
    req = requests.get(url, stream=True)
    content_length = req.headers.get('Content-Length')
    total = int(content_length) if content_length is not None else None
    progress = tqdm(unit="B", total=total)
    for chunk in req.iter_content(chunk_size=1024):
        if chunk: # filter out keep-alive new chunks
            progress.update(len(chunk))
            temp_file.write(chunk)
    progress.close()


def get_from_cache(url, cache_dir=None):
    """
    Given a URL, look for the corresponding dataset in the local cache.
    If it's not there, download it. Then return the path to the cached
 file.
    """
    if cache_dir is None:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        cache_dir = PYTORCH_PRETRAINED_BIGGAN_CACHE
    if sys.version_info[0] == 3 and isinstance(cache_dir, Path):
        cache_dir = str(cache_dir)

    if not os.path.exists(cache_dir):
        os.makedirs(cache_dir)

    # Get eTag to add to filename, if it exists.
    if url.startswith("s3://"):
        etag = s3_etag(url)
    else:
        response = requests.head(url, allow_redirects=True)
        if response.status_code != 200:
            raise IOError("HEAD request failed for url {} with status c
ode {}"
                          .format(url, response.status_code))
        etag = response.headers.get("ETag")

    filename = url_to_filename(url, etag)

    # get cache path to put the file
    cache_path = os.path.join(cache_dir, filename)

    if not os.path.exists(cache_path):
        # Download to temporary file, then copy to cache dir once finis
hed.
        # Otherwise you get corrupt cache entries if the download gets
 interrupted.
        with tempfile.NamedTemporaryFile() as temp_file:
            logger.info("%s not found in cache, downloading to %s", url
, temp_file.name)

            # GET file object
            if url.startswith("s3://"):
                s3_get(url, temp_file)
            else:
                http_get(url, temp_file)

            # we are copying the file before closing it, so flush to av
oid truncation
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
            temp_file.flush()
            # shutil.copyfileobj() starts at the current position, so g
o to the start
            temp_file.seek(0)

            logger.info("copying %s to cache at %s", temp_file.name, ca
che_path)
            with open(cache_path, 'wb') as cache_file:
                shutil.copyfileobj(temp_file, cache_file)

            logger.info("creating metadata file for %s", cache_path)
            meta = {'url': url, 'etag': etag}
            meta_path = cache_path + '.json'
            with open(meta_path, 'w', encoding="utf-8") as meta_file:
                json.dump(meta, meta_file)

            logger.info("removing temp file %s", temp_file.name)

    return cache_path


def read_set_from_file(filename):
    '''
    Extract a de-duped collection (set) of text from a file.
    Expected file format is one item per line.
    '''
    collection = set()
    with open(filename, 'r', encoding='utf-8') as file_:
        for line in file_:
            collection.add(line.rstrip())
    return collection


def get_file_extension(path, dot=True, lower=True):
    ext = os.path.splitext(path)[1]
    ext = ext if dot else ext[1:]
    return ext.lower() if lower else ext


PRETRAINED_MODEL_ARCHIVE_MAP = {
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    'biggan-deep-128': "https://s3.amazonaws.com/models.huggingface.co/
biggan/biggan-deep-128-pytorch_model.bin",
    'biggan-deep-256': "https://s3.amazonaws.com/models.huggingface.co/
biggan/biggan-deep-256-pytorch_model.bin",
    'biggan-deep-512': "https://s3.amazonaws.com/models.huggingface.co/
biggan/biggan-deep-512-pytorch_model.bin",
}

PRETRAINED_CONFIG_ARCHIVE_MAP = {
    'biggan-deep-128': "https://s3.amazonaws.com/models.huggingface.co/
biggan/biggan-deep-128-config.json",
    'biggan-deep-256': "https://s3.amazonaws.com/models.huggingface.co/
biggan/biggan-deep-256-config.json",
    'biggan-deep-512': "https://s3.amazonaws.com/models.huggingface.co/
biggan/biggan-deep-512-config.json",
}

WEIGHTS_NAME = 'pytorch_model.bin'
CONFIG_NAME = 'config.json'
import copy
import json

class BigGANConfig(object):
    """ Configuration class to store the configuration of a `BigGAN`.
        Defaults are for the 128x128 model.
        layers tuple are (up-sample in the layer ?, input channels, out
put channels)
    """
    def __init__(self,
                 output_dim=128,
                 z_dim=128,
                 class_embed_dim=128,
                 channel_width=128,
                 num_classes=1000,
                 layers=[(False, 16, 16),
                         (True, 16, 16),
                         (False, 16, 16),
                         (True, 16, 8),
                         (False, 8, 8),
                         (True, 8, 4),
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
                          (False, 4, 4),
                          (True, 4, 2),
                          (False, 2, 2),
                          (True, 2, 1)],
                 attention_layer_position=8,
                 eps=1e-4,
                 n_stats=51):
        """Constructs BigGANConfig. """
        self.output_dim = output_dim
        self.z_dim = z_dim
        self.class_embed_dim = class_embed_dim
        self.channel_width = channel_width
        self.num_classes = num_classes
        self.layers = layers
        self.attention_layer_position = attention_layer_position
        self.eps = eps
        self.n_stats = n_stats

    @classmethod
    def from_dict(cls, json_object):
        """Constructs a `BigGANConfig` from a Python dictionary of para
meters."""
        config = BigGANConfig()
        for key, value in json_object.items():
            config.__dict__[key] = value
        return config

    @classmethod
    def from_json_file(cls, json_file):
        """Constructs a `BigGANConfig` from a json file of parameter
s."""
        with open(json_file, "r", encoding='utf-8') as reader:
            text = reader.read()
        return cls.from_dict(json.loads(text))

    def __repr__(self):
        return str(self.to_json_string())

    def to_dict(self):
        """Serializes this instance to a Python dictionary."""
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        output = copy.deepcopy(self.__dict__)
        return output

    def to_json_string(self):
        """Serializes this instance to a JSON string."""
        return json.dumps(self.to_dict(), indent=2, sort_keys=True) + "
\n"




def snconv2d(eps=1e-12, **kwargs):
    return nn.utils.spectral_norm(nn.Conv2d(**kwargs), eps=eps)

def snlinear(eps=1e-12, **kwargs):
    return nn.utils.spectral_norm(nn.Linear(**kwargs), eps=eps)

def sn_embedding(eps=1e-12, **kwargs):
    return nn.utils.spectral_norm(nn.Embedding(**kwargs), eps=eps)

class SelfAttn(nn.Module):
    """ Self attention Layer"""
    def __init__(self, in_channels, eps=1e-12):
        super(SelfAttn, self).__init__()
        self.in_channels = in_channels
        self.snconv1x1_theta = snconv2d(in_channels=in_channels, out_ch
annels=in_channels//8,
                                        kernel_size=1, bias=False, eps=
eps)
        self.snconv1x1_phi = snconv2d(in_channels=in_channels, out_chan
nels=in_channels//8,
                                      kernel_size=1, bias=False, eps=ep
s)
        self.snconv1x1_g = snconv2d(in_channels=in_channels, out_channe
ls=in_channels//2,
                                    kernel_size=1, bias=False, eps=eps)
        self.snconv1x1_o_conv = snconv2d(in_channels=in_channels//2, ou
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
t_channels=in_channels,
                                              kernel_size=1, bias=False, eps
=eps)
        self.maxpool = nn.MaxPool2d(2, stride=2, padding=0)
        self.softmax  = nn.Softmax(dim=-1)
        self.gamma = nn.Parameter(torch.zeros(1))

    def forward(self, x):
        _, ch, h, w = x.size()
        # Theta path
        theta = self.snconv1x1_theta(x)
        theta = theta.view(-1, ch//8, h*w)
        # Phi path
        phi = self.snconv1x1_phi(x)
        phi = self.maxpool(phi)
        phi = phi.view(-1, ch//8, h*w//4)
        # Attn map
        attn = torch.bmm(theta.permute(0, 2, 1), phi)
        attn = self.softmax(attn)
        # g path
        g = self.snconv1x1_g(x)
        g = self.maxpool(g)
        g = g.view(-1, ch//2, h*w//4)
        # Attn_g - o_conv
        attn_g = torch.bmm(g, attn.permute(0, 2, 1))
        attn_g = attn_g.view(-1, ch//2, h, w)
        attn_g = self.snconv1x1_o_conv(attn_g)
        # Out
        out = x + self.gamma*attn_g
        return out


class BigGANBatchNorm(nn.Module):
    """ This is a batch norm module that can handle conditional input a
nd can be provided with pre-computed
        activation means and variances for various truncation parameter
s.
        We cannot just rely on torch.batch_norm since it cannot handle
        batched weights (pytorch 1.0.1). We computate batch_norm our-se
lf without updating running means and variances.
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        If you want to train this model you should add running means an
d variance computation logic.
    """
    def __init__(self, num_features, condition_vector_dim=None, n_stats
=51, eps=1e-4, conditional=True):
        super(BigGANBatchNorm, self).__init__()
        self.num_features = num_features
        self.eps = eps
        self.conditional = conditional

        # We use pre-computed statistics for n_stats values of truncati
on between 0 and 1
        self.register_buffer('running_means', torch.zeros(n_stats, num_
features))
        self.register_buffer('running_vars', torch.ones(n_stats, num_fe
atures))
        self.step_size = 1.0 / (n_stats - 1)

        if conditional:
            assert condition_vector_dim is not None
            self.scale = snlinear(in_features=condition_vector_dim, out
_features=num_features, bias=False, eps=eps)
            self.offset = snlinear(in_features=condition_vector_dim, ou
t_features=num_features, bias=False, eps=eps)
        else:
            self.weight = torch.nn.Parameter(torch.Tensor(num_features
))
            self.bias = torch.nn.Parameter(torch.Tensor(num_features))

    def forward(self, x, truncation, condition_vector=None):
        # Retreive pre-computed statistics associated to this truncatio
n
        coef, start_idx = math.modf(truncation / self.step_size)
        start_idx = int(start_idx)
        if coef != 0.0:  # Interpolate
            running_mean = self.running_means[start_idx] * coef + self.
running_means[start_idx + 1] * (1 - coef)
            running_var = self.running_vars[start_idx] * coef + self.ru
nning_vars[start_idx + 1] * (1 - coef)
        else:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
            running_mean = self.running_means[start_idx]
            running_var = self.running_vars[start_idx]

        if self.conditional:
            running_mean = running_mean.unsqueeze(0).unsqueeze(-1).unsq
ueeze(-1)
            running_var = running_var.unsqueeze(0).unsqueeze(-1).unsque
eze(-1)

            weight = 1 + self.scale(condition_vector).unsqueeze(-1).uns
queeze(-1)
            bias = self.offset(condition_vector).unsqueeze(-1).unsqueez
e(-1)

            out = (x - running_mean) / torch.sqrt(running_var + self.ep
s) * weight + bias
        else:
            out = F.batch_norm(x, running_mean, running_var, self.weigh
t, self.bias,
                               training=False, momentum=0.0, eps=self.e
ps)

        return out


class GenBlock(nn.Module):
    def __init__(self, in_size, out_size, condition_vector_dim, reducti
on_factor=4, up_sample=False,
                 n_stats=51, eps=1e-12):
        super(GenBlock, self).__init__()
        self.up_sample = up_sample
        self.drop_channels = (in_size != out_size)
        middle_size = in_size // reduction_factor

        self.bn_0 = BigGANBatchNorm(in_size, condition_vector_dim, n_st
ats=n_stats, eps=eps, conditional=True)
        self.conv_0 = snconv2d(in_channels=in_size, out_channels=middle
_size, kernel_size=1, eps=eps)

        self.bn_1 = BigGANBatchNorm(middle_size, condition_vector_dim,
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
n_stats=n_stats, eps=eps, conditional=True)
        self.conv_1 = snconv2d(in_channels=middle_size, out_channels=mi
ddle_size, kernel_size=3, padding=1, eps=eps)

        self.bn_2 = BigGANBatchNorm(middle_size, condition_vector_dim,
n_stats=n_stats, eps=eps, conditional=True)
        self.conv_2 = snconv2d(in_channels=middle_size, out_channels=mi
ddle_size, kernel_size=3, padding=1, eps=eps)

        self.bn_3 = BigGANBatchNorm(middle_size, condition_vector_dim,
n_stats=n_stats, eps=eps, conditional=True)
        self.conv_3 = snconv2d(in_channels=middle_size, out_channels=ou
t_size, kernel_size=1, eps=eps)

        self.relu = nn.ReLU()

    def forward(self, x, cond_vector, truncation):
        x0 = x

        x = self.bn_0(x, truncation, cond_vector)
        x = self.relu(x)
        x = self.conv_0(x)

        x = self.bn_1(x, truncation, cond_vector)
        x = self.relu(x)
        if self.up_sample:
            x = F.interpolate(x, scale_factor=2, mode='nearest')
        x = self.conv_1(x)

        x = self.bn_2(x, truncation, cond_vector)
        x = self.relu(x)
        x = self.conv_2(x)

        x = self.bn_3(x, truncation, cond_vector)
        x = self.relu(x)
        x = self.conv_3(x)

        if self.drop_channels:
            new_channels = x0.shape[1] // 2
            x0 = x0[:, :new_channels, ...]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        if self.up_sample:
            x0 = F.interpolate(x0, scale_factor=2, mode='nearest')

        out = x + x0
        return out

class Generator(nn.Module):
    def __init__(self, config):
        super(Generator, self).__init__()
        self.config = config
        ch = config.channel_width
        condition_vector_dim = config.z_dim * 2

        self.gen_z = snlinear(in_features=condition_vector_dim,
                              out_features=4 * 4 * 16 * ch, eps=config.
eps)

        layers = []
        for i, layer in enumerate(config.layers):
            if i == config.attention_layer_position:
                layers.append(SelfAttn(ch*layer[1], eps=config.eps))
            layers.append(GenBlock(ch*layer[1],
                                   ch*layer[2],
                                   condition_vector_dim,
                                   up_sample=layer[0],
                                   n_stats=config.n_stats,
                                   eps=config.eps))
        self.layers = nn.ModuleList(layers)

        self.bn = BigGANBatchNorm(ch, n_stats=config.n_stats, eps=confi
g.eps, conditional=False)
        self.relu = nn.ReLU()
        self.conv_to_rgb = snconv2d(in_channels=ch, out_channels=ch, ke
rnel_size=3, padding=1, eps=config.eps)
        self.tanh = nn.Tanh()

    def forward(self, cond_vector, truncation):
        z = self.gen_z(cond_vector[0].unsqueeze(0))

        # We use this conversion step to be able to use TF weights:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        # TF convention on shape is [batch, height, width, channels]
        # PT convention on shape is [batch, channels, height, width]
        z = z.view(-1, 4, 4, 16 * self.config.channel_width)
        z = z.permute(0, 3, 1, 2).contiguous()

        for i, layer in enumerate(self.layers):
            if isinstance(layer, GenBlock):
                z = layer(z, cond_vector[i+1].unsqueeze(0), truncation)
                # z = layer(z, cond_vector[].unsqueeze(0), truncation)
            else:
                z = layer(z)

        z = self.bn(z, truncation)
        z = self.relu(z)
        z = self.conv_to_rgb(z)
        z = z[:, :3, ...]
        z = self.tanh(z)
        return z

class BigGAN(nn.Module):
    """BigGAN Generator."""

    @classmethod
    def from_pretrained(cls, pretrained_model_name_or_path, cache_dir=N
one, *inputs, **kwargs):
        if pretrained_model_name_or_path in PRETRAINED_MODEL_ARCHIVE_MA
P:
            model_file = PRETRAINED_MODEL_ARCHIVE_MAP[pretrained_model_
name_or_path]
            config_file = PRETRAINED_CONFIG_ARCHIVE_MAP[pretrained_mode
l_name_or_path]
        else:
            model_file = os.path.join(pretrained_model_name_or_path, WE
IGHTS_NAME)
            config_file = os.path.join(pretrained_model_name_or_path, C
ONFIG_NAME)

        try:
            resolved_model_file = cached_path(model_file, cache_dir=cac
he_dir)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
            resolved_config_file = cached_path(config_file, cache_dir=c
ache_dir)
        except EnvironmentError:
            logger.error("Wrong model name, should be a valid path to a
folder containing "
                         "a {} file and a {} file or a model name in {}
".format(
                         WEIGHTS_NAME, CONFIG_NAME, PRETRAINED_MODEL_AR
CHIVE_MAP.keys()))
            raise

        logger.info("loading model {} from cache at {}".format(pretrain
ed_model_name_or_path, resolved_model_file))

        # Load config
        config = BigGANConfig.from_json_file(resolved_config_file)
        logger.info("Model config {}".format(config))

        # Instantiate model.
        model = cls(config, *inputs, **kwargs)
        state_dict = torch.load(resolved_model_file, map_location='cpu'
if not torch.cuda.is_available() else None)
        model.load_state_dict(state_dict, strict=False)
        return model

    def __init__(self, config):
        super(BigGAN, self).__init__()
        self.config = config
        self.embeddings = nn.Linear(config.num_classes, config.z_dim, b
ias=False)
        self.generator = Generator(config)

    def forward(self, z, class_label, truncation):
        assert 0 < truncation <= 1

        embed = self.embeddings(class_label)
        cond_vector = torch.cat((z, embed), dim=1)

        z = self.generator(cond_vector, truncation)
        return z
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
model = BigGAN.from_pretrained('biggan-deep-512')
model = model.cuda().eval()


def one_hot_from_int(int_or_list, batch_size=1):
    """ Create a one-hot vector from a class index or a list of class i
ndices.
        Params:
            int_or_list: int, or list of int, of the imagenet classes
 (between 0 and 999)
            batch_size: batch size.
                If int_or_list is an int create a batch of identical cl
asses.
                If int_or_list is a list, we should have `len(int_or_li
st) == batch_size`
        Output:
            array of shape (batch_size, 1000)
    """
    if isinstance(int_or_list, int):
        int_or_list = [int_or_list]

    if len(int_or_list) == 1 and batch_size > 1:
        int_or_list = [int_or_list[0]] * batch_size

    assert batch_size == len(int_or_list)

    array = np.zeros((batch_size, 1000), dtype=np.float32)
    for i, j in enumerate(int_or_list):
        array[i, j] = 1.0
    return array
```

```
Collecting pytorch-pretrained-biggan
  Downloading pytorch_pretrained_biggan-0.1.1-py3-none-any.whl (27 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dis
t-packages (from pytorch-pretrained-biggan) (2.23.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-p
ackages (from pytorch-pretrained-biggan) (1.19.5)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Collecting boto3
  Downloading boto3-1.17.17-py2.py3-none-any.whl (130 kB)
     |████████████████████████████████| 130 kB 26.0 MB/s
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-pa
ckages (from pytorch-pretrained-biggan) (4.41.1)
Requirement already satisfied: torch>=0.4.1 in /usr/local/lib/python3.
7/dist-packages (from pytorch-pretrained-biggan) (1.7.1+cu101)
Requirement already satisfied: typing-extensions in /usr/local/lib/pyth
on3.7/dist-packages (from torch>=0.4.1->pytorch-pretrained-biggan) (3.
7.4.3)
Collecting botocore<1.21.0,>=1.20.17
  Downloading botocore-1.20.17-py2.py3-none-any.whl (7.3 MB)
     |████████████████████████████████| 7.3 MB 14.7 MB/s
Collecting s3transfer<0.4.0,>=0.3.0
  Downloading s3transfer-0.3.4-py2.py3-none-any.whl (69 kB)
     |████████████████████████████████| 69 kB 8.9 MB/s
Collecting jmespath<1.0.0,>=0.7.1
  Downloading jmespath-0.10.0-py2.py3-none-any.whl (24 kB)
Collecting urllib3<1.27,>=1.25.4
  Downloading urllib3-1.26.3-py2.py3-none-any.whl (137 kB)
     |████████████████████████████████| 137 kB 62.4 MB/s
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/loca
l/lib/python3.7/dist-packages (from botocore<1.21.0,>=1.20.17->boto3->p
ytorch-pretrained-biggan) (2.8.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dis
t-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.21.0,>=1.20.17
->boto3->pytorch-pretrained-biggan) (1.15.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/pyt
hon3.7/dist-packages (from requests->pytorch-pretrained-biggan) (2020.1
2.5)
  Downloading urllib3-1.25.11-py2.py3-none-any.whl (127 kB)
     |████████████████████████████████| 127 kB 75.2 MB/s
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/pyth
on3.7/dist-packages (from requests->pytorch-pretrained-biggan) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.
7/dist-packages (from requests->pytorch-pretrained-biggan) (2.10)
Installing collected packages: urllib3, jmespath, botocore, s3transfer,
boto3, pytorch-pretrained-biggan
  Attempting uninstall: urllib3
    Found existing installation: urllib3 1.24.3
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    Uninstalling urllib3-1.24.3:
      Successfully uninstalled urllib3-1.24.3
ERROR: pip's dependency resolver does not currently take into account a
ll the packages that are installed. This behaviour is the source of the
following dependency conflicts.
datascience 0.10.6 requires folium==0.2.1, but you have folium 0.8.3 wh
ich is incompatible.
Successfully installed boto3-1.17.17 botocore-1.20.17 jmespath-0.10.0 p
ytorch-pretrained-biggan-0.1.1 s3transfer-0.3.4 urllib3-1.25.11
```

```
100%|██████████| 235773527/235773527 [00:03<00:00, 59171958.14B/s]
100%|██████████| 800/800 [00:00<00:00, 286986.25B/s]
```

In [ ]:

Latent coordinate

Choose a place to start in BigGAN (it'll be a dog. Probably a hound lol)

In [ ]:
```
class Pars(torch.nn.Module):
    def __init__(self):
        super(Pars, self).__init__()
        # params = torch.zeros(32, 1).normal_(std=1)

        # self.stds = torch.nn.Parameter(params)

        # params = torch.zeros(32, 1).normal_(std=.1)
        # self.means = torch.nn.Parameter(params)

        self.normu = torch.nn.Parameter(torch.zeros(32, 128).normal_(st
d=1).cuda())


        params_other = torch.zeros(32, 1000).normal_(-3.9, .3)
        self.cls = torch.nn.Parameter(params_other)

        self.thrsh_lat = torch.tensor(1).cuda()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        self.thrsh_cls = torch.tensor(1.9).cuda()



    def forward(self):
      # return self.ff2(self.ff1(self.latent_code)), torch.softmax(1000
*self.ff4(self.ff3(self.cls)), -1)
      return self.normu, torch.sigmoid(self.cls)


torch.manual_seed(0)


lats = Pars().cuda()
optimizer = torch.optim.Adam(lats.parameters(), .07)
eps = 0

nom = torchvision.transforms.Normalize((0.48145466, 0.4578275, 0.408210
73), (0.26862954, 0.26130258, 0.27577711))
t = perceptor.encode_text(tx.cuda()).detach().clone()

with torch.no_grad():
  al = (model(*lats(), 1).cpu()).numpy()
  print(lats())
  for allls in al:
    displ(allls)
    print('\n')
```

Out[ ]:
```
<torch._C.Generator at 0x7fedfbf9b3b0>
```

```
WARNING:root:Lossy conversion from float32 to uint8. Range [-0.97776532
17315674, 0.9921419620513916]. Convert image to uint8 prior to saving t
o suppress this warning.
```

```
(Parameter containing:
tensor([[-1.1258e+00, -1.1524e+00, -2.5058e-01,  ...,  1.1648e+00,
          9.2337e-01,  1.3873e+00],
        [-8.8338e-01, -4.1891e-01, -8.0483e-01,  ...,  1.4474e-01,
          1.9029e+00,  3.9040e-01],
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
                [-3.9373e-02, -8.0147e-01, -4.9554e-01,  ...,  5.5412e-01,
                 -1.8166e-01, -2.3447e-01],
                ...,
                [ 1.6598e+00,  1.7719e-01, -1.9280e-03,  ..., -1.7596e-01,
                  1.2778e+00,  1.0525e+00],
                [-3.3631e-01, -1.7877e-01, -9.9753e-01,  ..., -1.0989e+00,
                 -2.1547e-01,  3.3639e-01],
                [ 5.9011e-01, -8.3249e-01, -1.3715e+00,  ..., -3.0357e+00,
                 -1.7288e+00,  6.0198e-01]], device='cuda:0', requires_grad=Tru
        e), tensor([[0.0350, 0.0267, 0.0193,  ..., 0.0383, 0.0157, 0.0128],
                [0.0212, 0.0277, 0.0160,  ..., 0.0178, 0.0299, 0.0195],
                [0.0234, 0.0123, 0.0219,  ..., 0.0169, 0.0226, 0.0167],
                ...,
                [0.0148, 0.0184, 0.0141,  ..., 0.0266, 0.0226, 0.0160],
                [0.0309, 0.0207, 0.0247,  ..., 0.0204, 0.0239, 0.0276],
                [0.0229, 0.0219, 0.0253,  ..., 0.0170, 0.0183, 0.0198]],
               device='cuda:0'))
```

Out[ ]:

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Train

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Train and generate samples

In [ ]:

```
def checkin(loss):
  print(loss)
  best = torch.topk(loss[2], k=1, largest=False)[1]
  with torch.no_grad():
    al = model(*lats(), 1)[best:best+1].cpu().numpy()
  for allls in al:
    displ(allls)
    display.display(display.Image(str(3)+'.png'))
    print('\n')
  output.eval_js('new Audio("https://freesound.org/data/previews/80/80921_1022651-lq.ogg").play()')


def ascend_txt():
  out = model(*lats(), 1)

  cutn = 128
  p_s = []
  for ch in range(cutn):
    size = int(sideX*torch.zeros(1,).normal_(mean=.8, std=.3).clip(.5, .95))
    offsetx = torch.randint(0, sideX - size, ())
    offsety = torch.randint(0, sideX - size, ())
    apper = out[:, :, offsetx:offsetx + size, offsety:offsety + size]
    apper = torch.nn.functional.interpolate(apper, (224,224), mode='nearest')
    p_s.append(apper)
  into = torch.cat(p_s, 0)
  # into = torch.nn.functional.interpolate(out, (224,224), mode='nearest')

  into = nom((into + 1) / 2)

  iii = perceptor.encode_image(into)


  llls = lats()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
  lat_l = torch.abs(1 - torch.std(llls[0], dim=1)).mean() + \
  torch.abs(torch.mean(llls[0])).mean() + \
  4*torch.max(torch.square(llls[0]).mean(), lats.thrsh_lat)


  for array in llls[0]:
    mean = torch.mean(array)
    diffs = array - mean
    var = torch.mean(torch.pow(diffs, 2.0))
    std = torch.pow(var, 0.5)
    zscores = diffs / std
    skews = torch.mean(torch.pow(zscores, 3.0))
    kurtoses = torch.mean(torch.pow(zscores, 4.0)) - 3.0

    lat_l = lat_l + torch.abs(kurtoses) / llls[0].shape[0] + torch.abs(
skews) / llls[0].shape[0]




  cls_l = ((50*torch.topk(llls[1],largest=False,dim=1,k=999)[0])**2).me
an()




  return [lat_l, cls_l, -100*torch.cosine_similarity(t, iii, dim=-1).me
an()]

def train(epoch, i):
  loss1 = ascend_txt()
  loss = loss1[0] + loss1[1] + loss1[2]
  optimizer.zero_grad()
  loss.backward()
  optimizer.step()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
  if itt % 100 == 0:
    checkin(loss1)


itt = 0
for epochs in range(10000):
  for i in range(50000):
    train(eps, i)
    itt+=1
  eps+=1
```

```
WARNING:root:Lossy conversion from float32 to uint8. Range [-0.98546957
96966553, 0.9757171273231506]. Convert image to uint8 prior to saving t
o suppress this warning.
```

```
[tensor(4.5558, device='cuda:0', grad_fn=<AddBackward0>), tensor(1.154
2, device='cuda:0', grad_fn=<MeanBackward0>), tensor(-12.8516, device
='cuda:0', dtype=torch.float16, grad_fn=<MulBackward0>)]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
WARNING:root:Lossy conversion from float32 to uint8. Range [-0.99999994
03953552, 0.9999979138374329]. Convert image to uint8 prior to saving t
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
o suppress this warning.
```

```
[tensor(4.3895, device='cuda:0', grad_fn=<AddBackward0>), tensor(1.131
2, device='cuda:0', grad_fn=<MeanBackward0>), tensor(-36., device='cud
a:0', dtype=torch.float16, grad_fn=<MulBackward0>)]
```

```
WARNING:root:Lossy conversion from float32 to uint8. Range [-0.9999999
03953552, 0.9999991059303284]. Convert image to uint8 prior to saving t
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
o suppress this warning.
```

```
[tensor(4.3808, device='cuda:0', grad_fn=<AddBackward0>), tensor(0.808
6, device='cuda:0', grad_fn=<MeanBackward0>), tensor(-38.3750, device
='cuda:0', dtype=torch.float16, grad_fn=<MulBackward0>)]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
WARNING:root:Lossy conversion from float32 to uint8. Range [-0.99999582
76748657, 0.9999998211860657]. Convert image to uint8 prior to saving t

o suppress this warning.
```

```
[tensor(4.4302, device='cuda:0', grad_fn=<AddBackward0>), tensor(0.673
1, device='cuda:0', grad_fn=<MeanBackward0>), tensor(-39.7188, device
='cuda:0', dtype=torch.float16, grad_fn=<MulBackward0>)]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
WARNING:root:Lossy conversion from float32 to uint8. Range [-0.99998581
40945435, 0.9999997615814209]. Convert image to uint8 prior to saving t
o suppress this warning.
```

```
[tensor(4.4519, device='cuda:0', grad_fn=<AddBackward0>), tensor(0.621
6, device='cuda:0', grad_fn=<MeanBackward0>), tensor(-40.1562, device
='cuda:0', dtype=torch.float16, grad_fn=<MulBackward0>)]
```

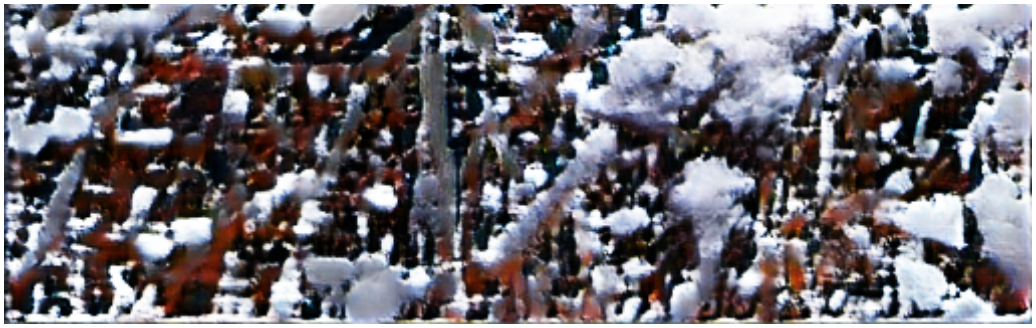
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
WARNING:root:Lossy conversion from float32 to uint8. Range [-0.99991828
20320129, 0.9999989867210388]. Convert image to uint8 prior to saving t
o suppress this warning.
```

```
[tensor(4.5007, device='cuda:0', grad_fn=<AddBackward0>), tensor(0.582
7, device='cuda:0', grad_fn=<MeanBackward0>), tensor(-40.5625, device
='cuda:0', dtype=torch.float16, grad_fn=<MulBackward0>)]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



```
WARNING:root:Lossy conversion from float32 to uint8. Range [-0.99993091
82167053, 0.9999998807907104]. Convert image to uint8 prior to saving t
o suppress this warning.
```

```
[tensor(4.5304, device='cuda:0', grad_fn=<AddBackward0>), tensor(0.557
1, device='cuda:0', grad_fn=<MeanBackward0>), tensor(-40.7812, device
='cuda:0', dtype=torch.float16, grad_fn=<MulBackward0>)]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
WARNING:root:Lossy conversion from float32 to uint8. Range [-0.99990934
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
13352966, 0.9999998807907104]. Convert image to uint8 prior to saving t
o suppress this warning.
```

```
[tensor(4.5485, device='cuda:0', grad_fn=<AddBackward0>), tensor(0.540
0, device='cuda:0', grad_fn=<MeanBackward0>), tensor(-40.6250, device
='cuda:0', dtype=torch.float16, grad_fn=<MulBackward0>)]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
WARNING:root:Lossy conversion from float32 to uint8. Range [-0.99991613
6264801, 0.9999988675117493]. Convert image to uint8 prior to saving to
suppress this warning.
```

```
[tensor(4.5588, device='cuda:0', grad_fn=<AddBackward0>), tensor(0.543
0, device='cuda:0', grad_fn=<MeanBackward0>), tensor(-42., device='cud
a:0', dtype=torch.float16, grad_fn=<MulBackward0>)]
```

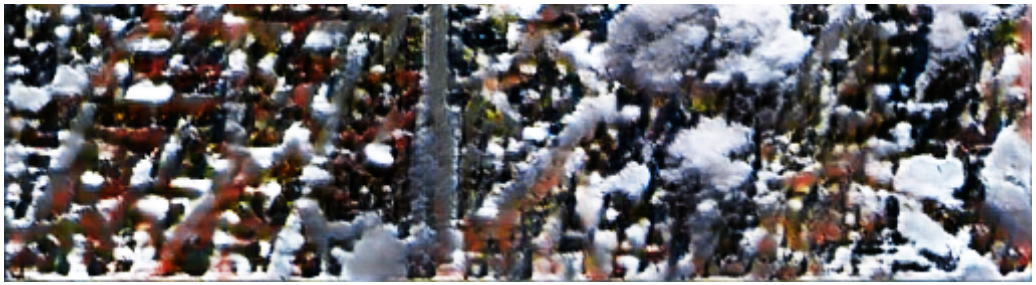
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
WARNING:root:Lossy conversion from float32 to uint8. Range [-0.9999060034751892, 0.9999990463256836]. Convert image to uint8 prior to saving to suppress this warning.
```

```
[tensor(4.5504, device='cuda:0', grad_fn=<AddBackward0>), tensor(0.5238, device='cuda:0', grad_fn=<MeanBackward0>), tensor(-42.3438, device='cuda:0', dtype=torch.float16, grad_fn=<MulBackward0>)]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



Diagnostics

If things don't work, try tweaking the learning rate. If that doesn't work, tweak the losses or even try changing the torch manual seed. The graphs below show the distributions of the class and latent vectors.

In [ ]:
```
with torch.no_grad():
  al = model(lats()[0], torch.zeros_like(lats()[1]), 1).cpu().numpy()
for allls in al:
  displ(allls)
  print('\n')

with torch.no_grad():
  al = model(torch.zeros_like(lats()[0]).normal_(), lats()[1], 1).cpu()
.numpy()
for allls in al:
  displ(allls)
  print('\n')


with torch.no_grad():
  these = tuple((lats()[0][:1].expand(32, -1), lats()[1][:1].expand(32,
-1)))
  al = model(*these, 1).cpu().numpy()
for allls in al:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
  displ(allls)
  print('\n')
```

In [ ]:
```
import matplotlib.pyplot as plt

with torch.no_grad():
  plt.close()
  th = plt.hist(lats()[0].detach().cpu().numpy().tolist())
  plt.show(th)
  plt.close()

  plt.show(plt.hist(lats()[1].detach().cpu().numpy().tolist()))
  print(torch.mean(lats()[0]))
  print(torch.mean(lats()[1]))
  print(lats()[0])
  print(lats()[1])
```

In [ ]:
```
with torch.no_grad():
  al = nom((1 + model(*lats(), 1).cpu()) / 2).numpy()
  for allls in al:
    displ(allls)
    print('\n')
```

Bottom

In [ ]:
```
tx = clip.tokenize("eros")
im = (torch.tensor(np.array(PIL.Image.open('/content/eros.png'))).unsqu
eeze(0)/255).permute(0,3,1,2)
im = (torch.nn.functional.interpolate(im, (224, 224)) - .45) / .22
perceptor(im.cuda(), tx.cuda())[0]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD