HW2 : CNN

7.1. From Fully Connected Layers to Convollutions

However, while we might be able to get away with one hundred thousand pixels, our hidden layer of size 1000 grossly underestimates the number of hidden units that it takes to learn good representations of images, so a practical system will still require billions of parameters. Moreover, learning a classifier by fitting so many parameters might require collecting an enormous dataset.

- Thus, we use CNN to avoid enormous parameters

7.1.1. Invariance

- We could sweep the image with a Waldo detector that could assign a score to each patch, indicating the likelihood that the patch contains Waldo. In fact, many object detection and segmentation algorithms are based on this approach (Long et al., 2015).
- We can find Invariance and then use it to figure out what image it is

7.1.2. Constraining the MLP

$[\mathbf{H}]_{i,j} = [\mathbf{U}]_{i,j} + \sum_k \sum_l [\mathbf{W}]_{i,j,k,l} [\mathbf{X}]_{k,l} = [\mathbf{U}]_{i,j} + \sum_a \sum_b [\mathbf{V}]_{i,j,a,b} [\mathbf{X}]_{i+a,j+b}$

7.1.2.1. Translation Invariance

$[\mathbf{H}]_{i,j} = u + \sum_a \sum_b [\mathbf{V}]_{a,b} [\mathbf{X}]_{i+a,j+b}$

- Need fewer coefficients and no longer depends on image

7.1.2.2. Locality

$[\mathbf{H}]_{i,j} = u + \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} [\mathbf{V}]_{a,b} [\mathbf{X}]_{i+a,j+b}.$

- This dramatic reduction in parameters brings us to our last desideratum, namely that deeper layers should represent larger and more complex aspects of an image. This can be achieved by interleaving nonlinearities and convolutional layers repeatedly.

7.1.3. Convolutions

- convolution between 2 functions

$$(f * g)(\mathbf{x}) = \int f(\mathbf{z})g(\mathbf{x} - \mathbf{z})d\mathbf{z}.$$

$$(f * g)(i) = \sum_a f(a)g(i - a).$$

$$(f * g)(i, j) = \sum_a \sum_b f(a, b)g(i - a, j - b).$$

7.1.4. Channels

$$[\mathsf{H}]_{i,j,d} = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} \sum_c [\mathsf{V}]_{a,b,c,d} [\mathsf{X}]_{i+a,j+b,c},$$

where d indexes the output channels in the hidden representations H. The subsequent convolutional layer will go on to take a third-order tensor,H, as input. We take (7.1.7), because of its generality, as the definition of a convolutional layer for multiple channels, where V is a kernel or filter of the layer.

7.2. Convolutions for Images

In [ ]:
```
pip install d2l
```

```
Collecting d2l
  Downloading d2l-1.0.3-py3-none-any.whl.metadata (556 bytes)
Collecting jupyter==1.0.0 (from d2l)
  Downloading jupyter-1.0.0-py2.py3-none-any.whl.metadata (995 bytes)
Collecting numpy==1.23.5 (from d2l)
  Downloading numpy-1.23.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014
_x86_64.whl.metadata (2.3 kB)
Collecting matplotlib==3.7.2 (from d2l)
  Downloading matplotlib-3.7.2-cp310-cp310-manylinux_2_17_x86_64.manylinux
2014_x86_64.whl.metadata (5.6 kB)
Collecting matplotlib-inline==0.1.6 (from d2l)
  Downloading matplotlib_inline-0.1.6-py3-none-any.whl.metadata (2.8 kB)
Collecting requests==2.31.0 (from d2l)
  Downloading requests-2.31.0-py3-none-any.whl.metadata (4.6 kB)
Collecting pandas==2.0.3 (from d2l)
  Downloading pandas-2.0.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014
_x86_64.whl.metadata (18 kB)
Collecting scipy==1.10.1 (from d2l)
  Downloading scipy-1.10.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014
_x86_64.whl.metadata (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.9/58.9 kB 1.4 MB/s eta 0:
00:00
Requirement already satisfied: notebook in /usr/local/lib/python3.10/dist-
packages (from jupyter==1.0.0->d2l) (6.5.5)
Collecting qtconsole (from jupyter==1.0.0->d2l)
  Downloading qtconsole-5.6.0-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: jupyter-console in /usr/local/lib/python3.1
0/dist-packages (from jupyter==1.0.0->d2l) (6.1.0)
Requirement already satisfied: nbconvert in /usr/local/lib/python3.10/dist
-packages (from jupyter==1.0.0->d2l) (6.5.4)
Requirement already satisfied: ipykernel in /usr/local/lib/python3.10/dist
-packages (from jupyter==1.0.0->d2l) (5.5.6)
Requirement already satisfied: ipywidgets in /usr/local/lib/python3.10/dis
t-packages (from jupyter==1.0.0->d2l) (7.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.
10/dist-packages (from matplotlib==3.7.2->d2l) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/d
ist-packages (from matplotlib==3.7.2->d2l) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python
3.10/dist-packages (from matplotlib==3.7.2->d2l) (4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python
3.10/dist-packages (from matplotlib==3.7.2->d2l) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib==3.7.2->d2l) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/
dist-packages (from matplotlib==3.7.2->d2l) (10.4.0)
Collecting pyparsing<3.1,>=2.3.1 (from matplotlib==3.7.2->d2l)
  Downloading pyparsing-3.0.9-py3-none-any.whl.metadata (4.2 kB)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/pyth
on3.10/dist-packages (from matplotlib==3.7.2->d2l) (2.8.2)
Requirement already satisfied: traitlets in /usr/local/lib/python3.10/dist
-packages (from matplotlib-inline==0.1.6->d2l) (5.7.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/d
ist-packages (from pandas==2.0.3->d2l) (2024.2)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.1
0/dist-packages (from pandas==2.0.3->d2l) (2024.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/
python3.10/dist-packages (from requests==2.31.0->d2l) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/d
ist-packages (from requests==2.31.0->d2l) (3.10)
```

```
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python
3.10/dist-packages (from requests==2.31.0->d2l) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.10/dist-packages (from requests==2.31.0->d2l) (2024.8.30)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib==3.7.2->d2l) (1.16.0)
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.
10/dist-packages (from ipykernel->jupyter==1.0.0->d2l) (0.2.0)
Requirement already satisfied: ipython>=5.0.0 in /usr/local/lib/python3.1
0/dist-packages (from ipykernel->jupyter==1.0.0->d2l) (7.34.0)
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.1
0/dist-packages (from ipykernel->jupyter==1.0.0->d2l) (6.1.12)
Requirement already satisfied: tornado>=4.2 in /usr/local/lib/python3.10/d
ist-packages (from ipykernel->jupyter==1.0.0->d2l) (6.3.3)
Requirement already satisfied: widgetsnbextension~=3.6.0 in /usr/local/li
b/python3.10/dist-packages (from ipywidgets->jupyter==1.0.0->d2l) (3.6.9)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/li
b/python3.10/dist-packages (from ipywidgets->jupyter==1.0.0->d2l) (3.0.13)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.
0 in /usr/local/lib/python3.10/dist-packages (from jupyter-console->jupyte
r==1.0.0->d2l) (3.0.48)
Requirement already satisfied: pygments in /usr/local/lib/python3.10/dist-
packages (from jupyter-console->jupyter==1.0.0->d2l) (2.18.0)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-pack
ages (from nbconvert->jupyter==1.0.0->d2l) (4.9.4)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.1
0/dist-packages (from nbconvert->jupyter==1.0.0->d2l) (4.12.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-pa
ckages (from nbconvert->jupyter==1.0.0->d2l) (6.1.0)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dis
t-packages (from nbconvert->jupyter==1.0.0->d2l) (0.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python
3.10/dist-packages (from nbconvert->jupyter==1.0.0->d2l) (0.4)
Requirement already satisfied: jinja2>=3.0 in /usr/local/lib/python3.10/di
st-packages (from nbconvert->jupyter==1.0.0->d2l) (3.1.4)
Requirement already satisfied: jupyter-core>=4.7 in /usr/local/lib/python
3.10/dist-packages (from nbconvert->jupyter==1.0.0->d2l) (5.7.2)
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/pytho
n3.10/dist-packages (from nbconvert->jupyter==1.0.0->d2l) (0.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.1
0/dist-packages (from nbconvert->jupyter==1.0.0->d2l) (2.1.5)
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python
3.10/dist-packages (from nbconvert->jupyter==1.0.0->d2l) (0.8.4)
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.1
0/dist-packages (from nbconvert->jupyter==1.0.0->d2l) (0.10.0)
Requirement already satisfied: nbformat>=5.1 in /usr/local/lib/python3.10/
dist-packages (from nbconvert->jupyter==1.0.0->d2l) (5.10.4)
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/pyth
on3.10/dist-packages (from nbconvert->jupyter==1.0.0->d2l) (1.5.1)
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.10/dist-
packages (from nbconvert->jupyter==1.0.0->d2l) (1.3.0)
Requirement already satisfied: pyzmq<25,>=17 in /usr/local/lib/python3.10/
dist-packages (from notebook->jupyter==1.0.0->d2l) (24.0.1)
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.10/di
st-packages (from notebook->jupyter==1.0.0->d2l) (23.1.0)
Requirement already satisfied: nest-asyncio>=1.5 in /usr/local/lib/python
3.10/dist-packages (from notebook->jupyter==1.0.0->d2l) (1.6.0)
Requirement already satisfied: Send2Trash>=1.8.0 in /usr/local/lib/python
3.10/dist-packages (from notebook->jupyter==1.0.0->d2l) (1.8.3)
Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.
```

```
10/dist-packages (from notebook->jupyter==1.0.0->d2l) (0.18.1)
Requirement already satisfied: prometheus-client in /usr/local/lib/python
3.10/dist-packages (from notebook->jupyter==1.0.0->d2l) (0.21.0)
Requirement already satisfied: nbclassic>=0.4.7 in /usr/local/lib/python3.
10/dist-packages (from notebook->jupyter==1.0.0->d2l) (1.1.0)
Collecting qtpy>=2.4.0 (from qtconsole->jupyter==1.0.0->d2l)
  Downloading QtPy-2.4.1-py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.
10/dist-packages (from ipython>=5.0.0->ipykernel->jupyter==1.0.0->d2l) (7
1.0.4)
Collecting jedi>=0.16 (from ipython>=5.0.0->ipykernel->jupyter==1.0.0->d2
l)
  Using cached jedi-0.19.1-py2.py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: decorator in /usr/local/lib/python3.10/dist
-packages (from ipython>=5.0.0->ipykernel->jupyter==1.0.0->d2l) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.10/di
st-packages (from ipython>=5.0.0->ipykernel->jupyter==1.0.0->d2l) (0.7.5)
Requirement already satisfied: backcall in /usr/local/lib/python3.10/dist-
packages (from ipython>=5.0.0->ipykernel->jupyter==1.0.0->d2l) (0.2.0)
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.10/di
st-packages (from ipython>=5.0.0->ipykernel->jupyter==1.0.0->d2l) (4.9.0)
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python
3.10/dist-packages (from jupyter-core>=4.7->nbconvert->jupyter==1.0.0->d2
l) (4.3.6)
Requirement already satisfied: notebook-shim>=0.2.3 in /usr/local/lib/pyth
on3.10/dist-packages (from nbclassic>=0.4.7->notebook->jupyter==1.0.0->d2
l) (0.2.4)
Requirement already satisfied: fastjsonschema>=2.15 in /usr/local/lib/pyth
on3.10/dist-packages (from nbformat>=5.1->nbconvert->jupyter==1.0.0->d2l)
(2.20.0)
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.1
0/dist-packages (from nbformat>=5.1->nbconvert->jupyter==1.0.0->d2l) (4.2
3.0)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.10/dist-p
ackages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->jupyter-consol
e->jupyter==1.0.0->d2l) (0.2.13)
Requirement already satisfied: ptyprocess in /usr/local/lib/python3.10/dis
t-packages (from terminado>=0.8.3->notebook->jupyter==1.0.0->d2l) (0.7.0)
Requirement already satisfied: argon2-cffi-bindings in /usr/local/lib/pyth
on3.10/dist-packages (from argon2-cffi->notebook->jupyter==1.0.0->d2l) (2
1.2.0)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/
dist-packages (from beautifulsoup4->nbconvert->jupyter==1.0.0->d2l) (2.6)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/d
ist-packages (from bleach->nbconvert->jupyter==1.0.0->d2l) (0.5.1)
Requirement already satisfied: parso<0.9.0,>=0.8.3 in /usr/local/lib/pytho
n3.10/dist-packages (from jedi>=0.16->ipython>=5.0.0->ipykernel->jupyter==
1.0.0->d2l) (0.8.4)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/
dist-packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert->jupyter==1.
0.0->d2l) (24.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /us
r/local/lib/python3.10/dist-packages (from jsonschema>=2.6->nbformat>=5.1-
>nbconvert->jupyter==1.0.0->d2l) (2023.12.1)
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/pytho
n3.10/dist-packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert->jupyt
er==1.0.0->d2l) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.1
0/dist-packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert->jupyter==
1.0.0->d2l) (0.20.0)
```

```
Requirement already satisfied: jupyter-server<3,>=1.8 in /usr/local/lib/py
thon3.10/dist-packages (from notebook-shim>=0.2.3->nbclassic>=0.4.7->noteb
ook->jupyter==1.0.0->d2l) (1.24.0)
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.10/di
st-packages (from argon2-cffi-bindings->argon2-cffi->notebook->jupyter==1.
0.0->d2l) (1.17.1)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist
-packages (from cffi>=1.0.1->argon2-cffi-bindings->argon2-cffi->notebook->
jupyter==1.0.0->d2l) (2.22)
Requirement already satisfied: anyio<4,>=3.1.0 in /usr/local/lib/python3.1
0/dist-packages (from jupyter-server<3,>=1.8->notebook-shim>=0.2.3->nbclas
sic>=0.4.7->notebook->jupyter==1.0.0->d2l) (3.7.1)
Requirement already satisfied: websocket-client in /usr/local/lib/python3.
10/dist-packages (from jupyter-server<3,>=1.8->notebook-shim>=0.2.3->nbcla
ssic>=0.4.7->notebook->jupyter==1.0.0->d2l) (1.8.0)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/d
ist-packages (from anyio<4,>=3.1.0->jupyter-server<3,>=1.8->notebook-shim>
=0.2.3->nbclassic>=0.4.7->notebook->jupyter==1.0.0->d2l) (1.3.1)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.1
0/dist-packages (from anyio<4,>=3.1.0->jupyter-server<3,>=1.8->notebook-sh
im>=0.2.3->nbclassic>=0.4.7->notebook->jupyter==1.0.0->d2l) (1.2.2)
Downloading d2l-1.0.3-py3-none-any.whl (111 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 111.7/111.7 kB 2.5 MB/s eta 0:
00:00
Downloading jupyter-1.0.0-py2.py3-none-any.whl (2.7 kB)
Downloading matplotlib-3.7.2-cp310-cp310-manylinux_2_17_x86_64.manylinux20
14_x86_64.whl (11.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.6/11.6 MB 63.4 MB/s eta 0:0
0:00
Downloading matplotlib_inline-0.1.6-py3-none-any.whl (9.4 kB)
Downloading numpy-1.23.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x
86_64.whl (17.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.1/17.1 MB 55.5 MB/s eta 0:0
0:00
Downloading pandas-2.0.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x
86_64.whl (12.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.3/12.3 MB 67.9 MB/s eta 0:0
0:00
Downloading requests-2.31.0-py3-none-any.whl (62 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.6/62.6 kB 2.7 MB/s eta 0:0
0:00
Downloading scipy-1.10.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x
86_64.whl (34.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 34.4/34.4 MB 14.1 MB/s eta 0:0
0:00
Downloading pyparsing-3.0.9-py3-none-any.whl (98 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.3/98.3 kB 4.1 MB/s eta 0:0
0:00
Downloading qtconsole-5.6.0-py3-none-any.whl (124 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 124.7/124.7 kB 2.0 MB/s eta 0:
00:00
Downloading QtPy-2.4.1-py3-none-any.whl (93 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 93.5/93.5 kB 2.1 MB/s eta 0:0
0:00
Using cached jedi-0.19.1-py2.py3-none-any.whl (1.6 MB)
Installing collected packages: requests, qtpy, pyparsing, numpy, matplotli
b-inline, jedi, scipy, pandas, matplotlib, qtconsole, jupyter, d2l
  Attempting uninstall: requests
    Found existing installation: requests 2.32.3
    Uninstalling requests-2.32.3:
```

```
      Successfully uninstalled requests-2.32.3
  Attempting uninstall: pyparsing
    Found existing installation: pyparsing 3.1.4
    Uninstalling pyparsing-3.1.4:
      Successfully uninstalled pyparsing-3.1.4
  Attempting uninstall: numpy
    Found existing installation: numpy 1.26.4
    Uninstalling numpy-1.26.4:
      Successfully uninstalled numpy-1.26.4
  Attempting uninstall: matplotlib-inline
    Found existing installation: matplotlib-inline 0.1.7
    Uninstalling matplotlib-inline-0.1.7:
      Successfully uninstalled matplotlib-inline-0.1.7
  Attempting uninstall: scipy
    Found existing installation: scipy 1.13.1
    Uninstalling scipy-1.13.1:
      Successfully uninstalled scipy-1.13.1
  Attempting uninstall: pandas
    Found existing installation: pandas 2.2.2
    Uninstalling pandas-2.2.2:
      Successfully uninstalled pandas-2.2.2
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.7.1
    Uninstalling matplotlib-3.7.1:
      Successfully uninstalled matplotlib-3.7.1
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the follo
wing dependency conflicts.
albucore 0.0.16 requires numpy>=1.24, but you have numpy 1.23.5 which is i
ncompatible.
albumentations 1.4.15 requires numpy>=1.24.4, but you have numpy 1.23.5 wh
ich is incompatible.
bigframes 1.21.0 requires numpy>=1.24.0, but you have numpy 1.23.5 which i
s incompatible.
chex 0.1.87 requires numpy>=1.24.1, but you have numpy 1.23.5 which is inc
ompatible.
google-colab 1.0.0 requires pandas==2.2.2, but you have pandas 2.0.3 which
is incompatible.
google-colab 1.0.0 requires requests==2.32.3, but you have requests 2.31.0
which is incompatible.
jax 0.4.33 requires numpy>=1.24, but you have numpy 1.23.5 which is incomp
atible.
jaxlib 0.4.33 requires numpy>=1.24, but you have numpy 1.23.5 which is inc
ompatible.
mizani 0.11.4 requires pandas>=2.1.0, but you have pandas 2.0.3 which is i
ncompatible.
plotnine 0.13.6 requires pandas<3.0.0,>=2.1.0, but you have pandas 2.0.3 w
hich is incompatible.
xarray 2024.9.0 requires numpy>=1.24, but you have numpy 1.23.5 which is i
ncompatible.
xarray 2024.9.0 requires pandas>=2.1, but you have pandas 2.0.3 which is i
ncompatible.
Successfully installed d2l-1.0.3 jedi-0.19.1 jupyter-1.0.0 matplotlib-3.7.
2 matplotlib-inline-0.1.6 numpy-1.23.5 pandas-2.0.3 pyparsing-3.0.9 qtcons
ole-5.6.0 qtpy-2.4.1 requests-2.31.0 scipy-1.10.1
```

In [ ]:

```
import torch
from torch import nn
from d2l import torch as d2l
```

7.2.1. The Cross-Correlation Operation

In [ ]:
```
def corr2d(X,K):
  h,w=K.shape
  Y=torch.zeros((X.shape[0]-h+1,X.shape[1]-w+1))
  for i in range(Y.shape[0]):
    for j in range(Y.shape[1]):
      Y[i,j]=(X[i:i+h,j:j+w]*K).sum()
  return Y
```

In [ ]:
```
X = torch.tensor([[0.0, 1.0, 2.0], [3.0, 4.0, 5.0], [6.0, 7.0, 8.0]])
K = torch.tensor([[0.0, 1.0], [2.0, 3.0]])
corr2d(X, K)
```

Out[ ]:
```
tensor([[19., 25.],
        [37., 43.]])
```

7.2.2. Convolutional Layers

In [ ]:
```
class Conv2D(nn.Module):
    def __init__(self, kernel_size):
        super().__init__()
        self.weight = nn.Parameter(torch.rand(kernel_size))
        self.bias = nn.Parameter(torch.zeros(1))

    def forward(self, x):
        return corr2d(x, self.weight) + self.bias
```

7.2.3. Object Edge Detection in Images

In [ ]:
```
X = torch.ones((6, 8))
X[:, 2:6] = 0
X
```

Out[ ]:
```
tensor([[1., 1., 0., 0., 0., 0., 1., 1.],
        [1., 1., 0., 0., 0., 0., 1., 1.],
        [1., 1., 0., 0., 0., 0., 1., 1.],
        [1., 1., 0., 0., 0., 0., 1., 1.],
        [1., 1., 0., 0., 0., 0., 1., 1.],
        [1., 1., 0., 0., 0., 0., 1., 1.]])
```

In [ ]:
```
K = torch.tensor([[1.0, -1.0]])
```

In [ ]:
```
Y = corr2d(X, K)
Y
```

Out[ ]:
```
tensor([[ 0.,  1.,  0.,  0.,  0., -1.,  0.],
        [ 0.,  1.,  0.,  0.,  0., -1.,  0.],
        [ 0.,  1.,  0.,  0.,  0., -1.,  0.],
        [ 0.,  1.,  0.,  0.,  0., -1.,  0.],
        [ 0.,  1.,  0.,  0.,  0., -1.,  0.],
        [ 0.,  1.,  0.,  0.,  0., -1.,  0.]])
```

In [ ]:
```
corr2d(X.t(), K)
```

Out[ ]:
```
tensor([[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]])
```

7.2.4. Learning a Kernel

In [ ]:
```
conv2d = nn.LazyConv2d(1, kernel_size=(1, 2), bias=False)

X = X.reshape((1, 1, 6, 8))
Y = Y.reshape((1, 1, 6, 7))
lr = 3e-2

for i in range(10):
    Y_hat = conv2d(X)
    l = (Y_hat - Y) ** 2
    conv2d.zero_grad()
    l.sum().backward()
    conv2d.weight.data[:] -= lr * conv2d.weight.grad
    if (i + 1) % 2 == 0:
        print(f'epoch {i + 1}, loss {l.sum():.3f}')
```

```
epoch 2, loss 7.900
epoch 4, loss 1.423
epoch 6, loss 0.279
epoch 8, loss 0.063
epoch 10, loss 0.017
```

In [ ]:
```
conv2d.weight.data.reshape((1, 2))
```

Out[ ]:
```
tensor([[ 0.9747, -0.9968]])
```

7.2.5. Cross-Correlation and Convolution

What if such layers perform strict convolution operations as defined in (7.1.6) instead of cross-correlations? In order to obtain the output of the strict convolution operation, we only need to flip the two-dimensional kernel tensor both horizontally and vertically, and then perform the cross-correlation operation with the input tensor.

Assuming that other conditions remain unchanged, the same output will be obtained

7.2.6. Feature Map and Receptive Field

As described in Section 7.1.4, the convolutional layer output in Fig. 7.2.1 is sometimes called a feature map, as it can be regarded as the learned representations (features) in the spatial dimensions (e.g., width and height) to the subsequent layer. In CNNs, for any element x of some layer, its receptive field refers to all the elements (from all the previous layers) that may affect the calculation of x during the forward propagation.

7.3. Padding and Stride

In [ ]:
```
import torch
from torch import nn
```

7.3.1. Padding

In [ ]:
```
def comp_conv2d(conv2d, X):
    X = X.reshape((1, 1) + X.shape)
    Y = conv2d(X)
    return Y.reshape(Y.shape[2:])

conv2d = nn.LazyConv2d(1, kernel_size=3, padding=1)
X = torch.rand(size=(8, 8))
comp_conv2d(conv2d, X).shape
```

Out[ ]:
```
torch.Size([8, 8])
```

In [ ]:
```
conv2d = nn.LazyConv2d(1, kernel_size=(5, 3), padding=(2, 1))
comp_conv2d(conv2d, X).shape
```

Out[ ]:
```
torch.Size([8, 8])
```

7.3.2. Stride

In [ ]:
```
conv2d = nn.LazyConv2d(1, kernel_size=3, padding=1, stride=2)
comp_conv2d(conv2d, X).shape
```

Out[ ]:
```
torch.Size([4, 4])
```

In [ ]:
```
conv2d = nn.LazyConv2d(1, kernel_size=(3, 5), padding=(0, 1), stride=(3,
comp_conv2d(conv2d, X).shape
```

Out[ ]:
```
torch.Size([2, 2])
```

7.4. Multiple Input and Multiple Output Channels

In [ ]:
```
import torch
from d2l import torch as d2l
```

7.4.1. Multiple Input Channels

In [ ]:
```
def corr2d_multi_in(X, K):
    return sum(d2l.corr2d(x, k) for x, k in zip(X, K))
```

In [ ]:
```
X = torch.tensor([[[0.0, 1.0, 2.0], [3.0, 4.0, 5.0], [6.0, 7.0, 8.0]],
               [[1.0, 2.0, 3.0], [4.0, 5.0, 6.0], [7.0, 8.0, 9.0]]])
K = torch.tensor([[[0.0, 1.0], [2.0, 3.0]], [[1.0, 2.0], [3.0, 4.0]]])

corr2d_multi_in(X, K)
```

Out[ ]:
```
tensor([[ 56.,  72.],
        [104., 120.]])
```

7.4.2. Multiple Output Channels

In [ ]:
```
def corr2d_multi_in_out(X, K):
    return torch.stack([corr2d_multi_in(X, k) for k in K], 0)
```

In [ ]:
```
K = torch.stack((K, K + 1, K + 2), 0)
K.shape
```

Out[ ]:
```
torch.Size([3, 2, 2, 2])
```

In [ ]:
```
corr2d_multi_in_out(X, K)
```

Out[ ]:
```
tensor([[[ 56.,  72.],
         [104., 120.]],

        [[ 76., 100.],
         [148., 172.]],

        [[ 96., 128.],
         [192., 224.]]])
```

7.4.3. 1x1 Convolutional Layer

In [ ]:
```
def corr2d_multi_in_out_1x1(X, K):
    c_i, h, w = X.shape
    c_o = K.shape[0]
    X = X.reshape((c_i, h * w))
    K = K.reshape((c_o, c_i))
    Y = torch.matmul(K, X)
    return Y.reshape((c_o, h, w))
```

In [ ]:
```
X = torch.normal(0, 1, (3, 3, 3))
K = torch.normal(0, 1, (2, 3, 1, 1))
Y1 = corr2d_multi_in_out_1x1(X, K)
Y2 = corr2d_multi_in_out(X, K)
assert float(torch.abs(Y1 - Y2).sum()) < 1e-6
```

7.5. Pooling

In [ ]:
```
import torch
from torch import nn
from d2l import torch as d2l
```

7.5.1. Maximum Pooling and Average Pooling

In [ ]:
```
def pool2d(X, pool_size, mode='max'):
    p_h, p_w = pool_size
    Y = torch.zeros((X.shape[0] - p_h + 1, X.shape[1] - p_w + 1))
    for i in range(Y.shape[0]):
        for j in range(Y.shape[1]):
            if mode == 'max':
```

```
                Y[i, j] = X[i: i + p_h, j: j + p_w].max()
            elif mode == 'avg':
                Y[i, j] = X[i: i + p_h, j: j + p_w].mean()
    return Y
```

In [ ]:
```
X = torch.tensor([[0.0, 1.0, 2.0], [3.0, 4.0, 5.0], [6.0, 7.0, 8.0]])
pool2d(X, (2, 2))
```

Out[ ]:
```
tensor([[4., 5.],
        [7., 8.]])
```

In [ ]:
```
pool2d(X, (2, 2), 'avg')
```

Out[ ]:
```
tensor([[2., 3.],
        [5., 6.]])
```

7.5.2. Padding and Stride

In [ ]:
```
X = torch.arange(16, dtype=torch.float32).reshape((1, 1, 4, 4))
X
```

Out[ ]:
```
tensor([[[[ 0.,  1.,  2.,  3.],
          [ 4.,  5.,  6.,  7.],
          [ 8.,  9., 10., 11.],
          [12., 13., 14., 15.]]]])
```

In [ ]:
```
pool2d = nn.MaxPool2d(3)
pool2d(X)
```

Out[ ]:
```
tensor([[[[10.]]]])
```

In [ ]:
```
pool2d = nn.MaxPool2d(3, padding=1, stride=2)
pool2d(X)
```

Out[ ]:
```
tensor([[[[ 5.,  7.],
          [13., 15.]]]])
```

In [ ]:
```
pool2d = nn.MaxPool2d((2, 3), stride=(2, 3), padding=(0, 1))
pool2d(X)
```

Out[ ]:
```
tensor([[[[ 5.,  7.],
          [13., 15.]]]])
```

7.5.3. Multiple Channels

In [ ]:
```
X = torch.cat((X, X + 1), 1)
X
```

Out[ ]:
```
tensor([[[[ 0.,  1.,  2.,  3.],
          [ 4.,  5.,  6.,  7.],
          [ 8.,  9., 10., 11.],
          [12., 13., 14., 15.]],

         [[ 1.,  2.,  3.,  4.],
          [ 5.,  6.,  7.,  8.],
          [ 9., 10., 11., 12.],
          [13., 14., 15., 16.]],

         [[ 1.,  2.,  3.,  4.],
          [ 5.,  6.,  7.,  8.],
          [ 9., 10., 11., 12.],
          [13., 14., 15., 16.]],

         [[ 2.,  3.,  4.,  5.],
          [ 6.,  7.,  8.,  9.],
          [10., 11., 12., 13.],
          [14., 15., 16., 17.]]]])
```

In [ ]:
```
pool2d = nn.MaxPool2d(3, padding=1, stride=2)
pool2d(X)
```

Out[ ]:
```
tensor([[[[ 5.,  7.],
          [13., 15.]],

         [[ 6.,  8.],
          [14., 16.]]]])
```

7.6. Convolutional Neural Networks (LeNet)

In [ ]:
```
import torch
from torch import nn
from d2l import torch as d2l
```

7.6.1. LeNet

In [ ]:
```
def init_cnn(module):
    if type(module) == nn.Linear or type(module) == nn.Conv2d:
        nn.init.xavier_uniform_(module.weight)

class LeNet(d2l.Classifier):
    def __init__(self, lr=0.1, num_classes=10):
        super().__init__()
        self.save_hyperparameters()
        self.net = nn.Sequential(
            nn.LazyConv2d(6, kernel_size=5, padding=2), nn.Sigmoid(),
            nn.AvgPool2d(kernel_size=2, stride=2),
            nn.LazyConv2d(16, kernel_size=5), nn.Sigmoid(),
            nn.AvgPool2d(kernel_size=2, stride=2),
            nn.Flatten(),
            nn.LazyLinear(120), nn.Sigmoid(),
            nn.LazyLinear(84), nn.Sigmoid(),
            nn.LazyLinear(num_classes))
```

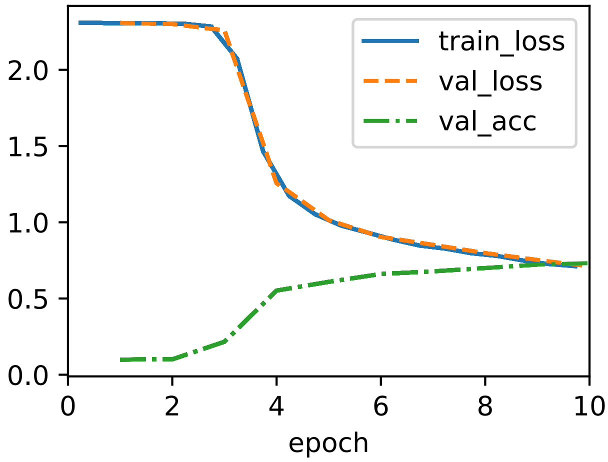
In [ ]:
```
@d2l.add_to_class(d2l.Classifier)
def layer_summary(self, X_shape):
```

```
    X = torch.randn(*X_shape)
    for layer in self.net:
        X = layer(X)
        print(layer.__class__.__name__, 'output shape:\t', X.shape)

model = LeNet()
model.layer_summary((1, 1, 28, 28))
```

```
Conv2d output shape:     torch.Size([1, 6, 28, 28])
Sigmoid output shape:    torch.Size([1, 6, 28, 28])
AvgPool2d output shape:  torch.Size([1, 6, 14, 14])
Conv2d output shape:     torch.Size([1, 16, 10, 10])
Sigmoid output shape:    torch.Size([1, 16, 10, 10])
AvgPool2d output shape:  torch.Size([1, 16, 5, 5])
Flatten output shape:    torch.Size([1, 400])
Linear output shape:     torch.Size([1, 120])
Sigmoid output shape:    torch.Size([1, 120])
Linear output shape:     torch.Size([1, 84])
Sigmoid output shape:    torch.Size([1, 84])
Linear output shape:     torch.Size([1, 10])
```

7.6.2. Training

In [ ]:
```
trainer = d2l.Trainer(max_epochs=10, num_gpus=1)
data = d2l.FashionMNIST(batch_size=128)
model = LeNet(lr=0.1)
model.apply_init([next(iter(data.get_dataloader(True)))[0]], init_cnn)
trainer.fit(model, data)
```



8.2. Networks Using Blocks (VGG)

In [ ]:
```
import torch
from torch import nn
from d2l import torch as d2l
```

8.2.1. VGG Blocks

In [ ]:
```
def vgg_block(num_convs, out_channels):
    layers = []
    for _ in range(num_convs):
        layers.append(nn.LazyConv2d(out_channels, kernel_size=3, padding=
```

```
        layers.append(nn.ReLU())
    layers.append(nn.MaxPool2d(kernel_size=2,stride=2))
    return nn.Sequential(*layers)
```

8.2.2. VGG Network

In [ ]:
```
class VGG(d2l.Classifier):
    def __init__(self, arch, lr=0.1, num_classes=10):
        super().__init__()
        self.save_hyperparameters()
        conv_blks = []
        for (num_convs, out_channels) in arch:
            conv_blks.append(vgg_block(num_convs, out_channels))
        self.net = nn.Sequential(
            *conv_blks, nn.Flatten(),
            nn.LazyLinear(4096), nn.ReLU(), nn.Dropout(0.5),
            nn.LazyLinear(4096), nn.ReLU(), nn.Dropout(0.5),
            nn.LazyLinear(num_classes))
        self.net.apply(d2l.init_cnn)
```

In [ ]:
```
VGG(arch=((1, 64), (1, 128), (2, 256), (2, 512), (2, 512))).layer_summary
    (1, 1, 224, 224))
```

```
Sequential output shape:         torch.Size([1, 64, 112, 112])
Sequential output shape:         torch.Size([1, 128, 56, 56])
Sequential output shape:         torch.Size([1, 256, 28, 28])
Sequential output shape:         torch.Size([1, 512, 14, 14])
Sequential output shape:         torch.Size([1, 512, 7, 7])
Flatten output shape:    torch.Size([1, 25088])
Linear output shape:     torch.Size([1, 4096])
ReLU output shape:       torch.Size([1, 4096])
Dropout output shape:    torch.Size([1, 4096])
Linear output shape:     torch.Size([1, 4096])
ReLU output shape:       torch.Size([1, 4096])
Dropout output shape:    torch.Size([1, 4096])
Linear output shape:     torch.Size([1, 10])
```

8.2.3. Training

- I tried more than 50 minutes to operate this cell, but colab shows a message that the access to the runtime is lost and it stops operating. I assume that this training process takes too much time and colab automatically stops this cell.

In [ ]:
```
model = VGG(arch=((1, 16), (1, 32), (2, 64), (2, 128), (2, 128)), lr=0.01
trainer = d2l.Trainer(max_epochs=10, num_gpus=1)
data = d2l.FashionMNIST(batch_size=128, resize=(224, 224))
model.apply_init([next(iter(data.get_dataloader(True)))[0]], d2l.init_cnn
trainer.fit(model, data)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-6-8d3dfa6be93b> in <cell line: 4>()
      2 trainer = d2l.Trainer(max_epochs=10, num_gpus=1)
      3 data = d2l.FashionMNIST(batch_size=128, resize=(224, 224))
----> 4 model.apply_init([next(iter(data.get_dataloader(True)))[0]], d2l.init_cnn)
      5 trainer.fit(model, data)

/usr/local/lib/python3.10/dist-packages/d2l/torch.py in apply_init(self, inputs, init)
    228     def apply_init(self, inputs, init=None):
    229         """Defined in :numref:`sec_lazy_init`"""
--> 230         self.forward(*inputs)
    231         if init is not None:
    232             self.net.apply(init)

/usr/local/lib/python3.10/dist-packages/d2l/torch.py in forward(self, X)
    191     def forward(self, X):
    192         assert hasattr(self, 'net'), 'Neural network is defined'
--> 193         return self.net(X)
    194 
    195     def plot(self, key, value, train):

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py in _wrapped_call_impl(self, *args, **kwargs)
   1551             return self._compiled_call_impl(*args, **kwargs)  # type: ignore[misc]
   1552         else:
-> 1553             return self._call_impl(*args, **kwargs)
   1554 
   1555     def _call_impl(self, *args, **kwargs):

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py in _call_impl(self, *args, **kwargs)
   1560                 or _global_backward_pre_hooks or _global_backward_hooks
   1561                 or _global_forward_hooks or _global_forward_pre_hooks):
-> 1562             return forward_call(*args, **kwargs)
   1563 
   1564         try:

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/container.py in forward(self, input)
    217     def forward(self, input):
    218         for module in self:
--> 219             input = module(input)
    220         return input
    221 

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py in _wrapped_call_impl(self, *args, **kwargs)
   1551             return self._compiled_call_impl(*args, **kwargs)  # type: ignore[misc]
   1552         else:
-> 1553             return self._call_impl(*args, **kwargs)
   1554 
```

```
   1555     def _call_impl(self, *args, **kwargs):

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py in _cal
l_impl(self, *args, **kwargs)
   1560                 or _global_backward_pre_hooks or _global_backward_
hooks
   1561                 or _global_forward_hooks or _global_forward_pre_ho
oks):
-> 1562             return forward_call(*args, **kwargs)
   1563 
   1564         try:

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/container.py in f
orward(self, input)
    217     def forward(self, input):
    218         for module in self:
--> 219             input = module(input)
    220         return input
    221 

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py in _wra
pped_call_impl(self, *args, **kwargs)
   1551             return self._compiled_call_impl(*args, **kwargs)  # ty
pe: ignore[misc]
   1552         else:
-> 1553             return self._call_impl(*args, **kwargs)
   1554 
   1555     def _call_impl(self, *args, **kwargs):

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py in _cal
l_impl(self, *args, **kwargs)
   1601                 args = bw_hook.setup_input_hook(args)
   1602 
-> 1603             result = forward_call(*args, **kwargs)
   1604             if _global_forward_hooks or self._forward_hooks:
   1605                 for hook_id, hook in (

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/conv.py in forwar
d(self, input)
    456 
    457     def forward(self, input: Tensor) -> Tensor:
--> 458         return self._conv_forward(input, self.weight, self.bias)
    459 
    460 class Conv3d(_ConvNd):

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/conv.py in _conv_
forward(self, input, weight, bias)
    452                             weight, bias, self.stride,
    453                             _pair(0), self.dilation, self.groups)
--> 454         return F.conv2d(input, weight, bias, self.stride,
    455                         self.padding, self.dilation, self.groups)
    456 

KeyboardInterrupt: 
```

8.6. Residual Networks (ResNet) and ResNeXt

In [ ]:
```
import torch
from torch import nn
```

```
from torch.nn import functional as F
from d2l import torch as d2l
```

8.6.1. Function Classes

$$f_{\mathcal{F}}^* \stackrel{\mathrm{def}}{=} \mathop{\mathrm{argmin}}_f L(\mathbf{X}, \mathbf{y}, f) \text{ subject to } f \in \mathcal{F}.$$

8.6.2. Residual Blocks

In [ ]:
```
class Residual(nn.Module):
    def __init__(self, num_channels, use_1x1conv=False, strides=1):
        super().__init__()
        self.conv1 = nn.LazyConv2d(num_channels, kernel_size=3, padding=1
                                   stride=strides)
        self.conv2 = nn.LazyConv2d(num_channels, kernel_size=3, padding=1
        if use_1x1conv:
            self.conv3 = nn.LazyConv2d(num_channels, kernel_size=1,
                                       stride=strides)
        else:
            self.conv3 = None
        self.bn1 = nn.LazyBatchNorm2d()
        self.bn2 = nn.LazyBatchNorm2d()

    def forward(self, X):
        Y = F.relu(self.bn1(self.conv1(X)))
        Y = self.bn2(self.conv2(Y))
        if self.conv3:
            X = self.conv3(X)
        Y += X
        return F.relu(Y)
```

In [ ]:
```
blk = Residual(3)
X = torch.randn(4, 3, 6, 6)
blk(X).shape
```

Out[ ]:
```
torch.Size([4, 3, 6, 6])
```

In [ ]:
```
blk = Residual(6, use_1x1conv=True, strides=2)
blk(X).shape
```

Out[ ]:
```
torch.Size([4, 6, 3, 3])
```

8.6.3. ResNet Model

In [ ]:
```
class ResNet(d2l.Classifier):
    def b1(self):
        return nn.Sequential(
            nn.LazyConv2d(64, kernel_size=7, stride=2, padding=3),
            nn.LazyBatchNorm2d(), nn.ReLU(),
            nn.MaxPool2d(kernel_size=3, stride=2, padding=1))
```

In [ ]:
```
@d2l.add_to_class(ResNet)
def block(self, num_residuals, num_channels, first_block=False):
    blk = []
    for i in range(num_residuals):
```

```
        if i == 0 and not first_block:
            blk.append(Residual(num_channels, use_1x1conv=True, strides=2
        else:
            blk.append(Residual(num_channels))
    return nn.Sequential(*blk)
```

In [ ]:
```
@d2l.add_to_class(ResNet)
def __init__(self, arch, lr=0.1, num_classes=10):
    super(ResNet, self).__init__()
    self.save_hyperparameters()
    self.net = nn.Sequential(self.b1())
    for i, b in enumerate(arch):
        self.net.add_module(f'b{i+2}', self.block(*b, first_block=(i==0))
    self.net.add_module('last', nn.Sequential(
        nn.AdaptiveAvgPool2d((1, 1)), nn.Flatten(),
        nn.LazyLinear(num_classes)))
    self.net.apply(d2l.init_cnn)
```

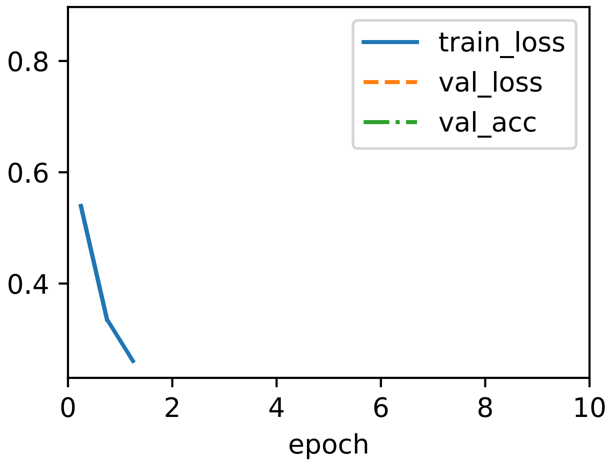
In [ ]:
```
class ResNet18(ResNet):
    def __init__(self, lr=0.1, num_classes=10):
        super().__init__(((2, 64), (2, 128), (2, 256), (2, 512)),
                       lr, num_classes)

ResNet18().layer_summary((1, 1, 96, 96))
```

```
Sequential output shape:	 torch.Size([1, 64, 24, 24])
Sequential output shape:	 torch.Size([1, 64, 24, 24])
Sequential output shape:	 torch.Size([1, 128, 12, 12])
Sequential output shape:	 torch.Size([1, 256, 6, 6])
Sequential output shape:	 torch.Size([1, 512, 3, 3])
Sequential output shape:	 torch.Size([1, 10])
```

8.6.3. Training

This also doesn't work. colab shows it cant access to runtime process and the process just stops. Below is the image after I tried this cell for 55 minutes.

In [ ]:
```
model = ResNet18(lr=0.01)
trainer = d2l.Trainer(max_epochs=10, num_gpus=1)
data = d2l.FashionMNIST(batch_size=128, resize=(96, 96))
model.apply_init([next(iter(data.get_dataloader(True)))[0]], d2l.init_cnn
trainer.fit(model, data)
```



Discussions and Exerciese

7.1.5. Discussion questions

In the context of adding more channels to CNNs,how does increasing the dimensionality of input data impact the computational complexity of the model?

- Adding more channels can expand the computations. Therfore it can gain complexity. However, this needs more computational power and can cause overfitting problem.

7.2.8. Exerciese

In [ ]:
```
# apply edge detecting at diagonal matrix
import torch
X = torch.eye(6)
X
```

Out[ ]:
```
tensor([[1., 0., 0., 0., 0., 0.],
        [0., 1., 0., 0., 0., 0.],
        [0., 0., 1., 0., 0., 0.],
        [0., 0., 0., 1., 0., 0.],
        [0., 0., 0., 0., 1., 0.],
        [0., 0., 0., 0., 0., 1.]])
```

In [ ]:
```
K = torch.tensor([[1.0, -1.0]])
```

In [ ]:
```
Y = corr2d(X, K)
Y
```

Out[ ]:
```
tensor([[ 1.,  0.,  0.,  0.,  0.],
        [-1.,  1.,  0.,  0.,  0.],
        [ 0., -1.,  1.,  0.,  0.],
        [ 0.,  0., -1.,  1.,  0.],
        [ 0.,  0.,  0., -1.,  1.],
        [ 0.,  0.,  0.,  0., -1.]])
```

In [ ]:
```
corr2d(X.t(), K)
```

Out[ ]:
```
tensor([[ 1.,  0.,  0.,  0.,  0.],
        [-1.,  1.,  0.,  0.,  0.],
        [ 0., -1.,  1.,  0.,  0.],
        [ 0.,  0., -1.,  1.,  0.],
        [ 0.,  0.,  0., -1.,  1.],
        [ 0.,  0.,  0.,  0., -1.]])
```

When applied to a diagoal matrix, it still can detect edge. Also it can detect when the matrix is transposed. Since the black-white and white-black edge is same, there appears two line of 1s and -1s

7.3.4. Discussion question

What are the computational benefits of a stride larger than 1?

- A stride larger than 1 reduces the number of operations required to compute the output. This leads to fewer convolution operations and results faster computation. Also using large stride can reduce memory usage

What might be statistical benefits of a stride larger than 1?

- Using large stride can increase the receptive field of each neuron in the network. This makes the network can capture more global features.

7.4.4. Discussion question

What are the advantages and trade-offs of using multiple channels in CNNs, particularly in terms of balancing parameter reduction and model expressiveness?

- Using multiple channels in CNNs allows capturing diverse and complex features. However, the cost of computing convolutions will increase linearly with the number of channels.

7.5.5. Exercise

Maxpooling Vs Avg Pooling

- Used code made by Chatgpt

In [ ]:
```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
import torch.nn.functional as F
from torch.utils.data import DataLoader

# 데이터 전처리: 이미지를 Tensor로 변환하고 정규화
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

# CIFAR-10 데이터셋 로드
trainset = torchvision.datasets.CIFAR10(root='./data', train=True, downlo
trainloader = DataLoader(trainset, batch_size=100, shuffle=True)

testset = torchvision.datasets.CIFAR10(root='./data', train=False, downlo
testloader = DataLoader(testset, batch_size=100, shuffle=False)
```

```
Files already downloaded and verified
Files already downloaded and verified
```

In [ ]:
```
# MaxPooling을 사용하는 CNN 모델
class CNN_MaxPool(nn.Module):
    def __init__(self):
```

```
        super(CNN_MaxPool, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, 3, padding=1)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
        self.fc1 = nn.Linear(64 * 8 * 8, 512)
        self.fc2 = nn.Linear(512, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 64 * 8 * 8)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# Average Pooling을 사용하는 CNN 모델
class CNN_AvgPool(nn.Module):
    def __init__(self):
        super(CNN_AvgPool, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, 3, padding=1)
        self.pool = nn.AvgPool2d(2, 2)
        self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
        self.fc1 = nn.Linear(64 * 8 * 8, 512)
        self.fc2 = nn.Linear(512, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 64 * 8 * 8)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

In [ ]:

```
# 훈련 함수
def train(model, trainloader, criterion, optimizer, device):
    model.train()
    running_loss = 0.0
    for images, labels in trainloader:
        images, labels = images.to(device), labels.to(device)

        # 그래디언트 초기화
        optimizer.zero_grad()

        # 순전파, 손실 계산, 역전파, 최적화
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()

    return running_loss / len(trainloader)

# 테스트 함수
def test(model, testloader, device):
    model.eval()
    correct = 0
    total = 0
    with torch.no_grad():
        for images, labels in testloader:
```

```
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    return 100 * correct / total
```

In [ ]:
```
# GPU 또는 CPU 사용 설정
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# MaxPooling 모델 설정
model_maxpool = CNN_MaxPool().to(device)
criterion = nn.CrossEntropyLoss()
optimizer_maxpool = optim.Adam(model_maxpool.parameters(), lr=0.001)

# Average Pooling 모델 설정
model_avgpool = CNN_AvgPool().to(device)
optimizer_avgpool = optim.Adam(model_avgpool.parameters(), lr=0.001)

# 모델 훈련 및 평가
epochs = 5
for epoch in range(epochs):
    print(f"Epoch {epoch+1}/{epochs}:\n")

    # MaxPooling 모델 훈련 및 테스트
    loss_maxpool = train(model_maxpool, trainloader, criterion, optimizer
    accuracy_maxpool = test(model_maxpool, testloader, device)
    print(f"MaxPooling - Loss: {loss_maxpool:.4f}, Accuracy: {accuracy_ma

    # Average Pooling 모델 훈련 및 테스트
    loss_avgpool = train(model_avgpool, trainloader, criterion, optimizer
    accuracy_avgpool = test(model_avgpool, testloader, device)
    print(f"Average Pooling - Loss: {loss_avgpool:.4f}, Accuracy: {accura
```

```
Epoch 1/5:

MaxPooling - Loss: 1.3261, Accuracy: 58.79%
Average Pooling - Loss: 1.3966, Accuracy: 60.18%

Epoch 2/5:

MaxPooling - Loss: 0.9465, Accuracy: 69.17%
Average Pooling - Loss: 1.0390, Accuracy: 65.20%

Epoch 3/5:

MaxPooling - Loss: 0.7625, Accuracy: 71.72%
Average Pooling - Loss: 0.8687, Accuracy: 67.64%

Epoch 4/5:

MaxPooling - Loss: 0.6029, Accuracy: 72.26%
Average Pooling - Loss: 0.7255, Accuracy: 70.42%

Epoch 5/5:

MaxPooling - Loss: 0.4547, Accuracy: 72.33%
Average Pooling - Loss: 0.5970, Accuracy: 71.65%
```

In training image data, we can see MaxPooling shows better performance.

7.6.4. Exercises

- Change AvgPool to MaxPool
- Change sigmoid to Relu

In [ ]:
```
import torch
from torch import nn
from d2l import torch as d2l
```

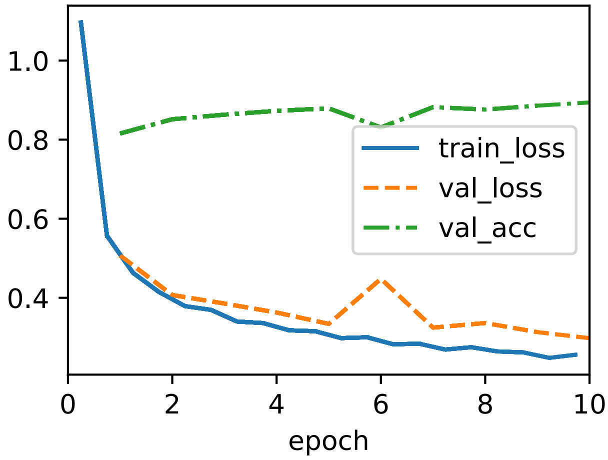
In [ ]:
```
def init_cnn(module):
    if type(module) == nn.Linear or type(module) == nn.Conv2d:
        nn.init.xavier_uniform_(module.weight)

class LeNet(d2l.Classifier):
    def __init__(self, lr=0.1, num_classes=10):
        super().__init__()
        self.save_hyperparameters()
        self.net = nn.Sequential(
            nn.LazyConv2d(6, kernel_size=5, padding=2), nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
            nn.LazyConv2d(16, kernel_size=5), nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
            nn.Flatten(),
            nn.LazyLinear(120), nn.ReLU(),
            nn.LazyLinear(84), nn.ReLU(),
            nn.LazyLinear(num_classes))
```

In [ ]:
```
@d2l.add_to_class(d2l.Classifier)
def layer_summary(self, X_shape):
    X = torch.randn(*X_shape)
    for layer in self.net:
        X = layer(X)
        print(layer.__class__.__name__, 'output shape:\t', X.shape)

model = LeNet()
model.layer_summary((1, 1, 28, 28))
```

```
Conv2d output shape:     torch.Size([1, 6, 28, 28])
ReLU output shape:       torch.Size([1, 6, 28, 28])
MaxPool2d output shape:  torch.Size([1, 6, 14, 14])
Conv2d output shape:     torch.Size([1, 16, 10, 10])
ReLU output shape:       torch.Size([1, 16, 10, 10])
MaxPool2d output shape:  torch.Size([1, 16, 5, 5])
Flatten output shape:    torch.Size([1, 400])
Linear output shape:     torch.Size([1, 120])
ReLU output shape:       torch.Size([1, 120])
Linear output shape:     torch.Size([1, 84])
ReLU output shape:       torch.Size([1, 84])
Linear output shape:     torch.Size([1, 10])
```

In [ ]:
```
trainer = d2l.Trainer(max_epochs=10, num_gpus=1)
data = d2l.FashionMNIST(batch_size=128)
model = LeNet(lr=0.1)
```

```
model.apply_init([next(iter(data.get_dataloader(True)))[0]], init_cnn)
trainer.fit(model, data)
```



8.2.5. Discussion question

AlexNet vs VGG

- num of parameter

:AlexNet : 60million, VGG : 138million

- AlexNet has 5 convolutional layers and 3 fully connected layers.
- VGG has 13 convolutional layers and 3 fully connected layers

Both has many parameter and needs high computational power

To reduce this computational cost, we can use Global Average Pooling. this can do pooling process of the feature maps into a single value per feature map. Therfore, we can reduce computational cost

In [ ]:

```

```