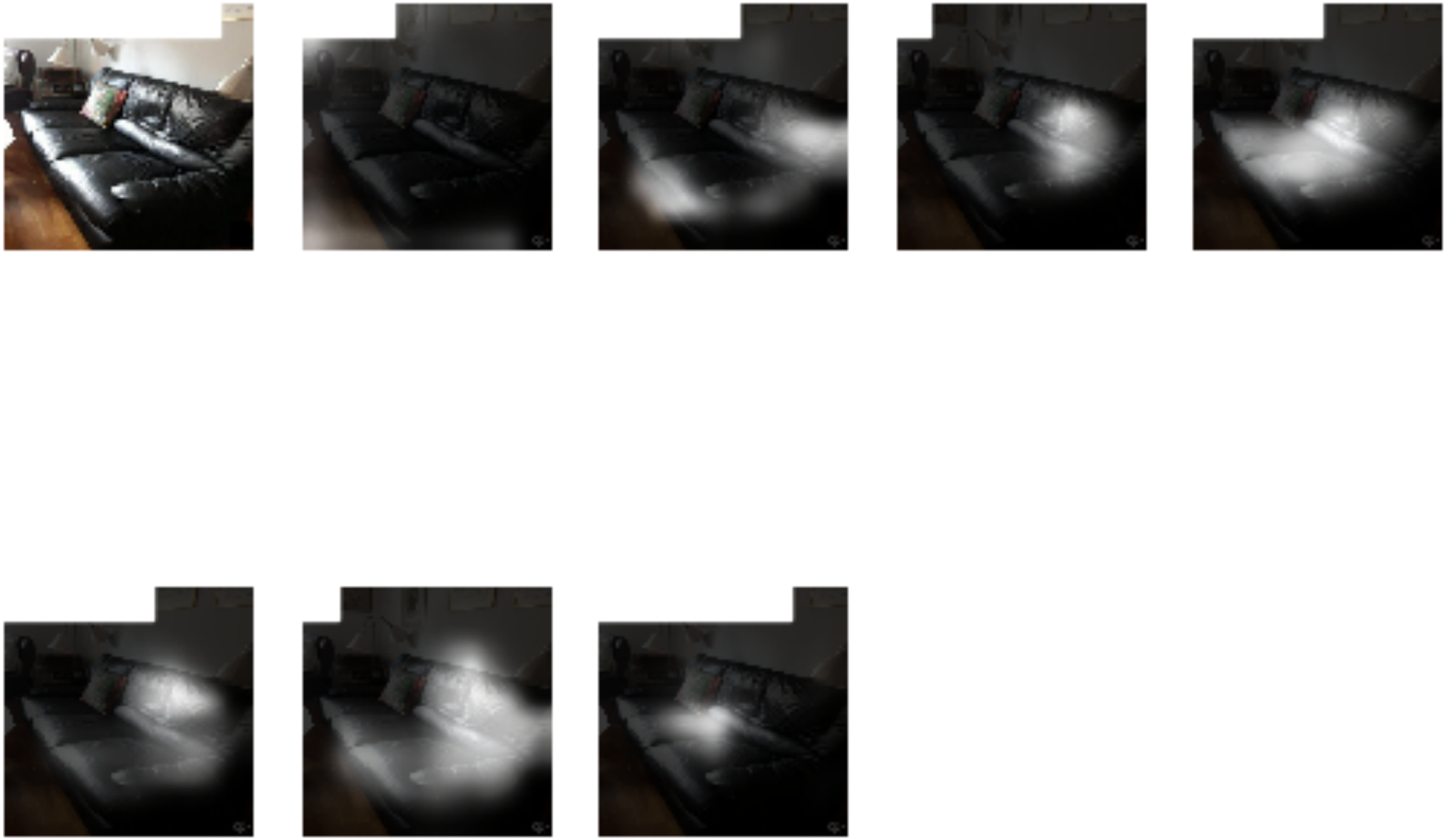
Guide to your own **artificial intelligence** application in 3 easy steps

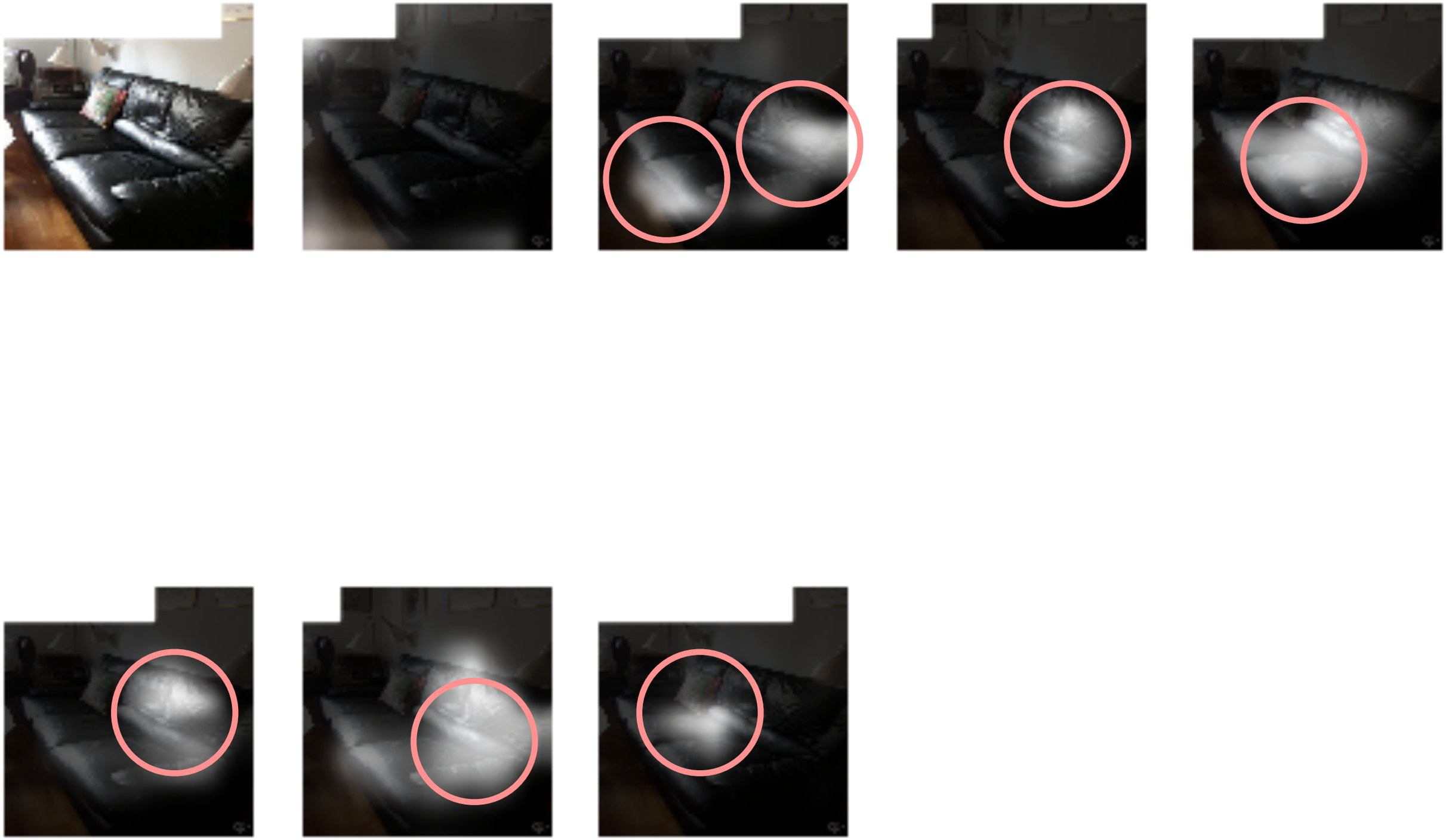
Norah Klintberg Sakal

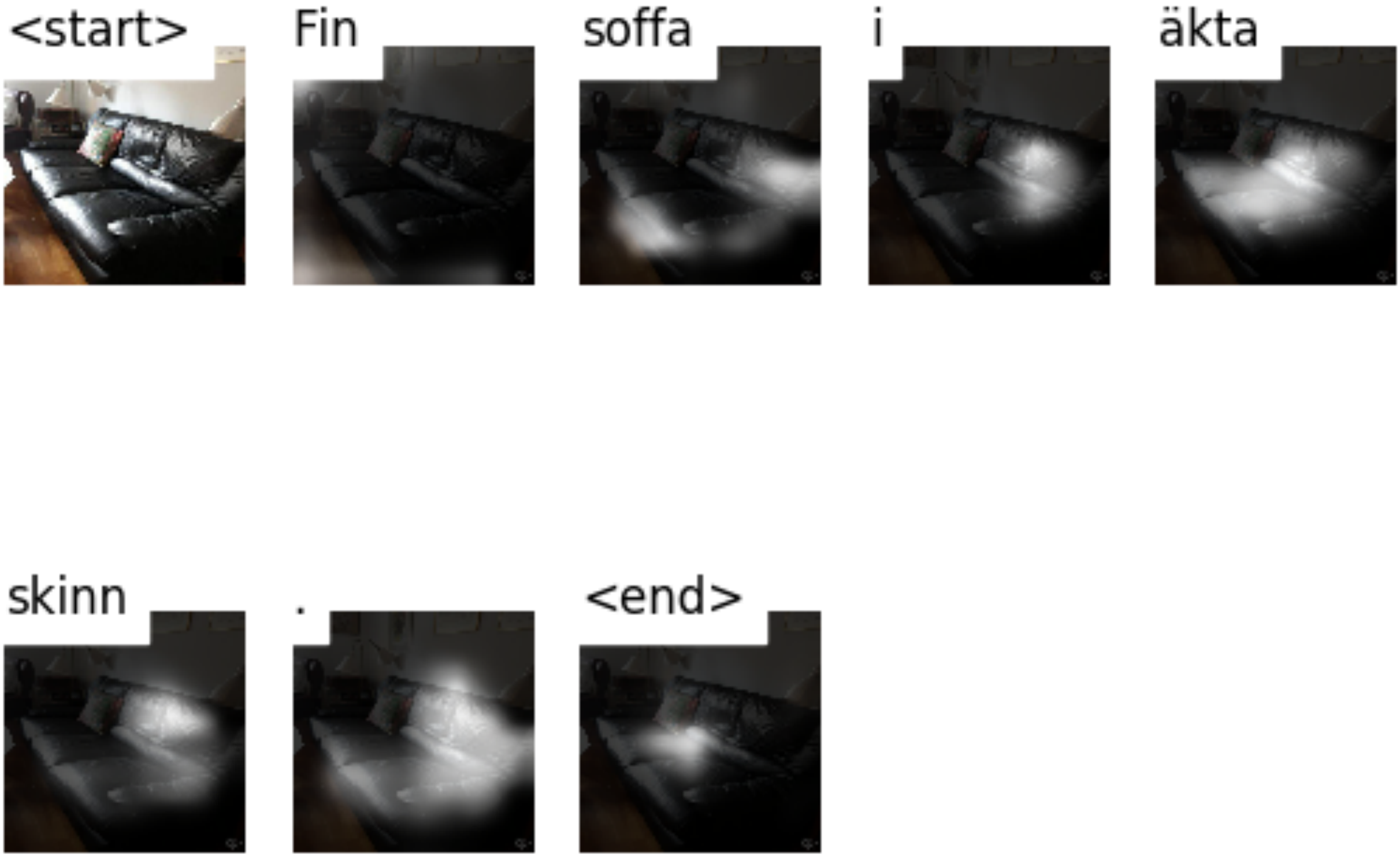
Norah Klintberg Sakal

Data scientist

Artificial intelligence + ads






<start>
Fin
soffa
i
äkta
skinn
.
<end>

“Nice couch in genuine leather”

“We are selling our **nice** and **comfortable** couch because we are **moving”**

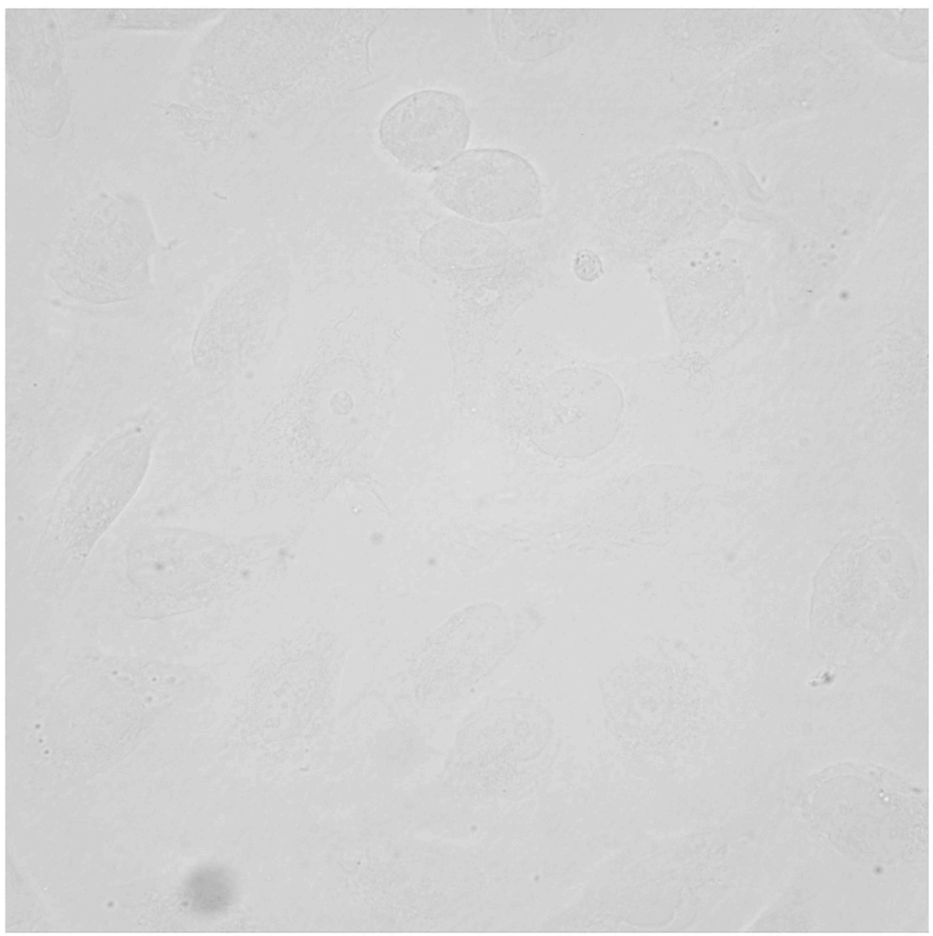
?

“We are selling our nice and comfortable couch because we are moving”

Cancer cell counter

1. Upload an image for nuclei count

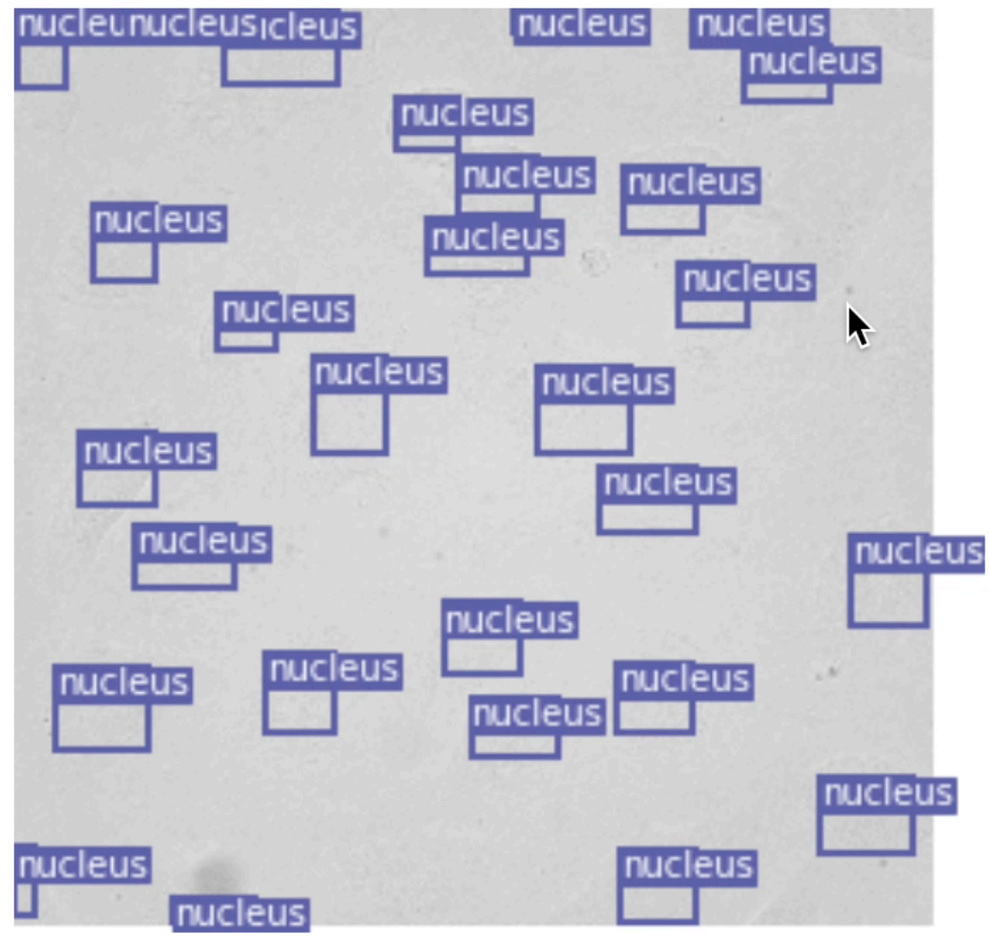
Input image



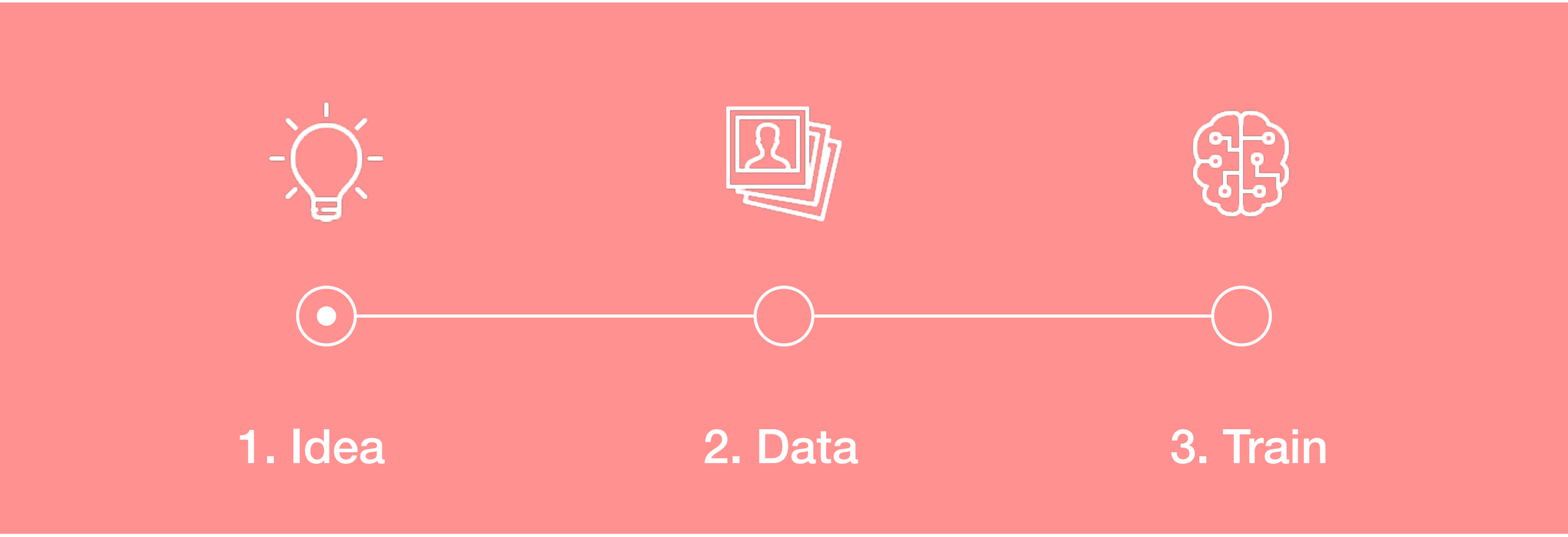
SELECT NEW IMAGE

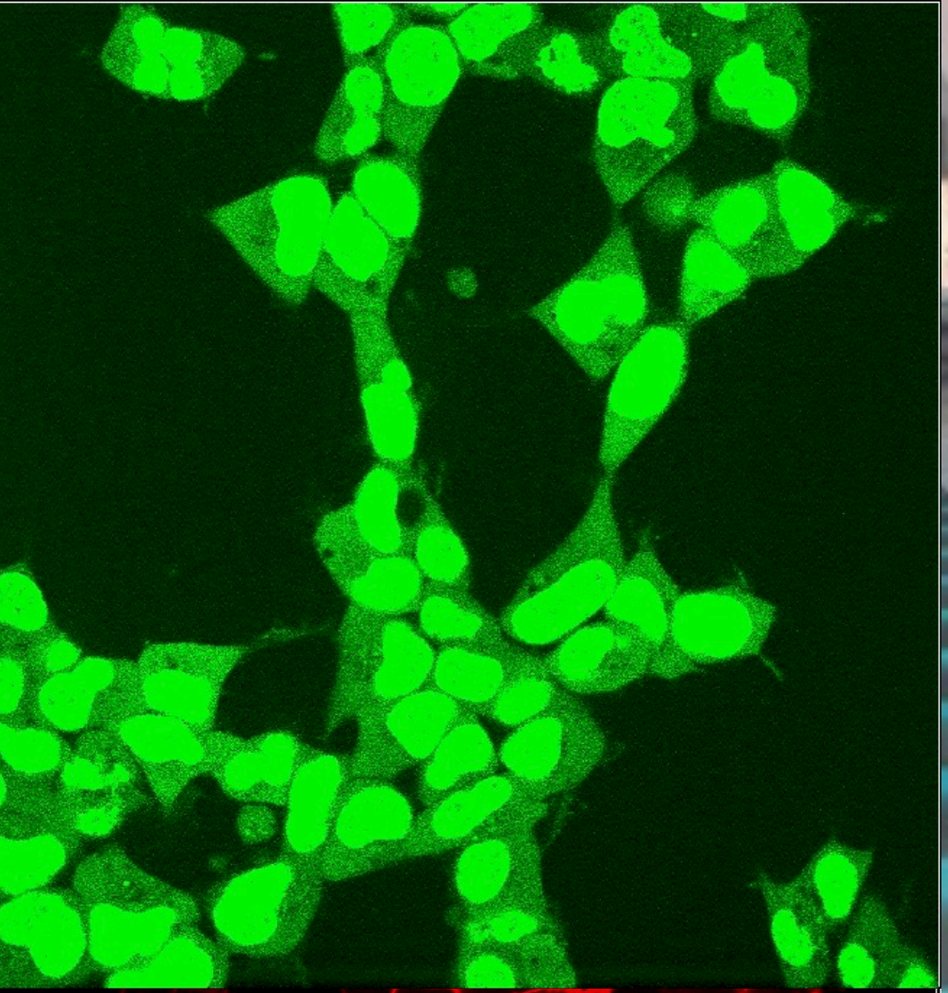
2. Predicted image: **brightfield**

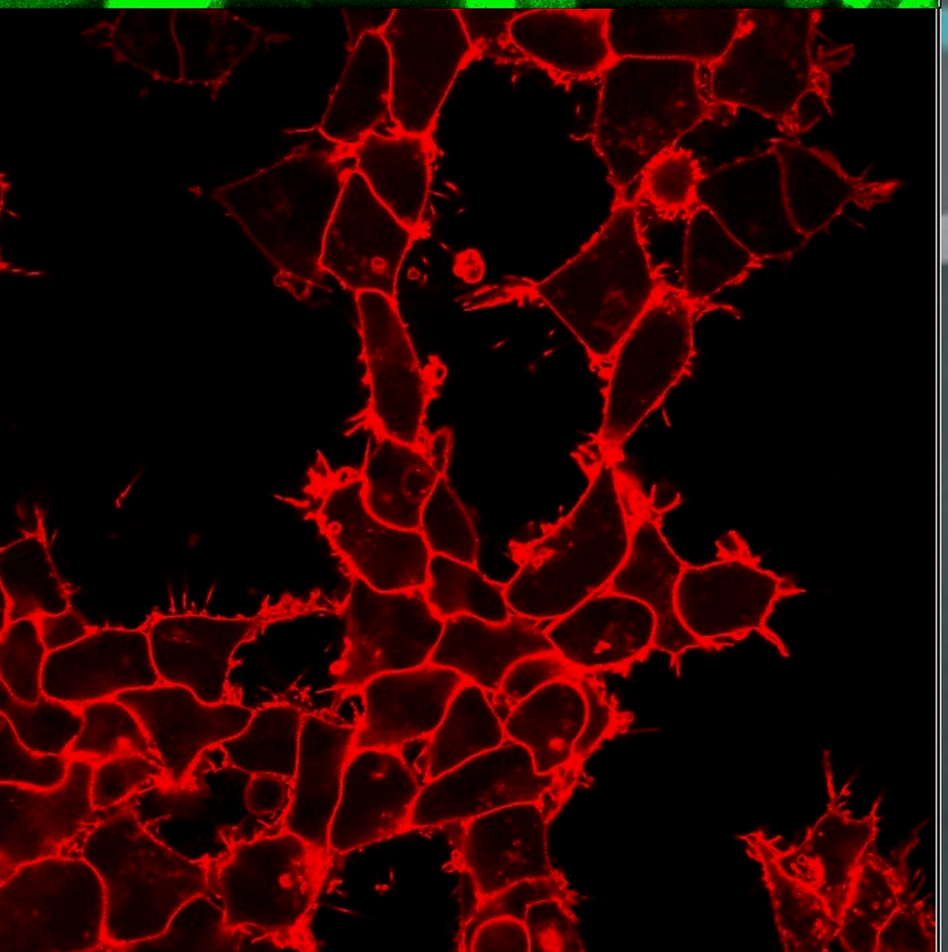
The model predicted: **28 nuclei**

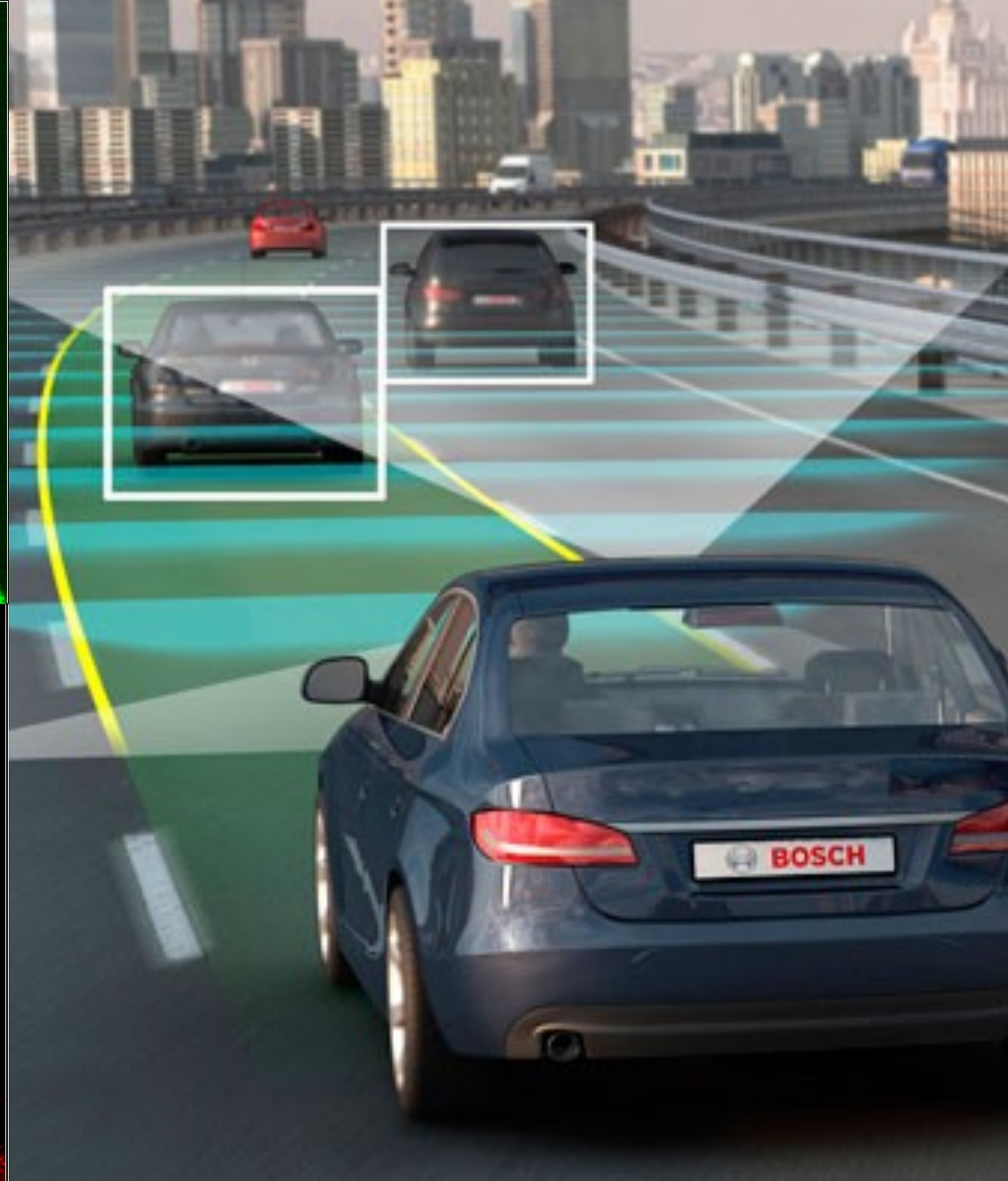


Outline 3 steps








BOSCH

1. Find a **topic**

Passion

Day to day problem

Trends

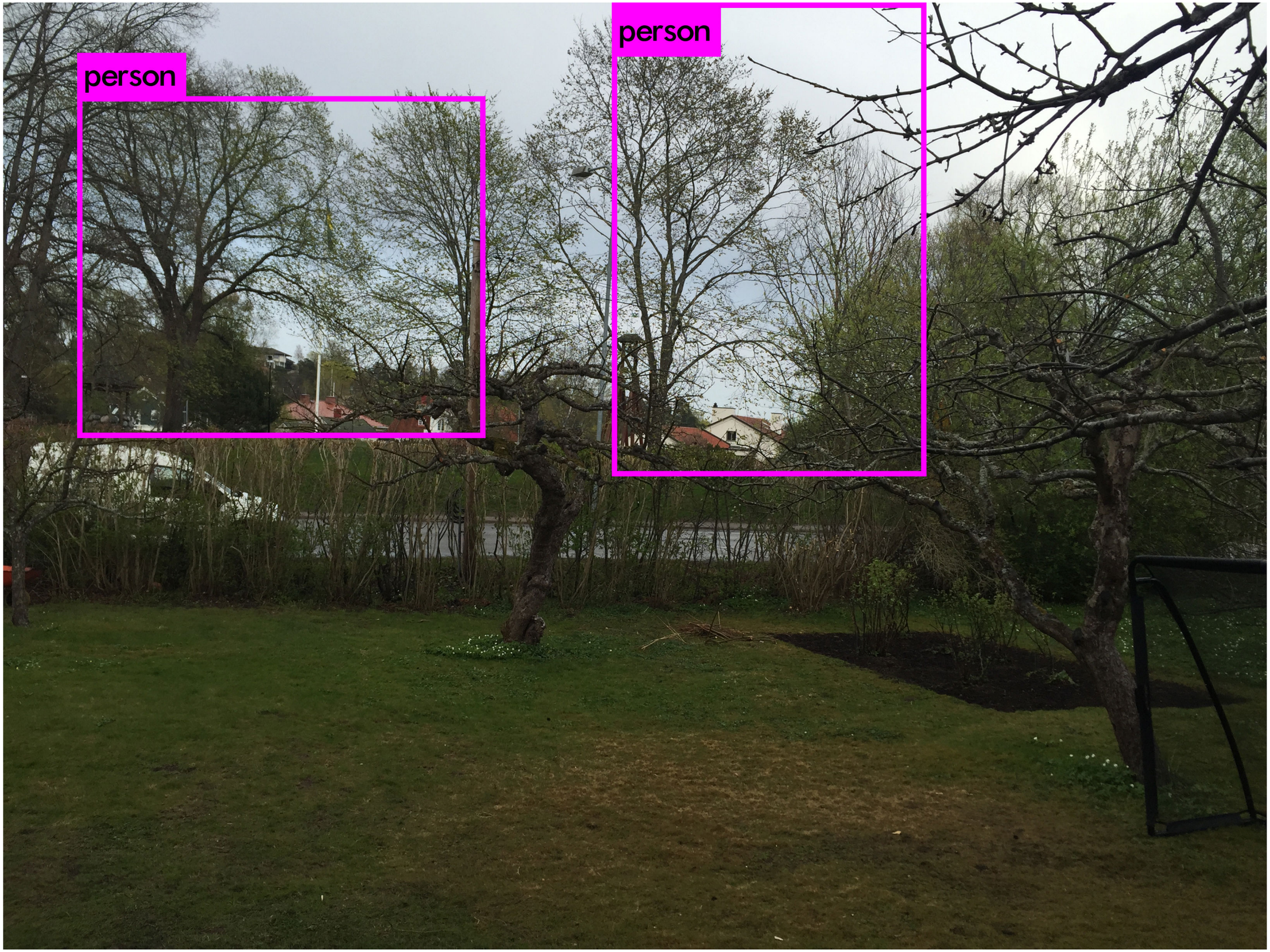
1. Find a **topic**

Passion

Day to day problem

Trends

“Don’t really know why we are selling it, but we were renewing so here we go”


person
person

1. Find a **topic**

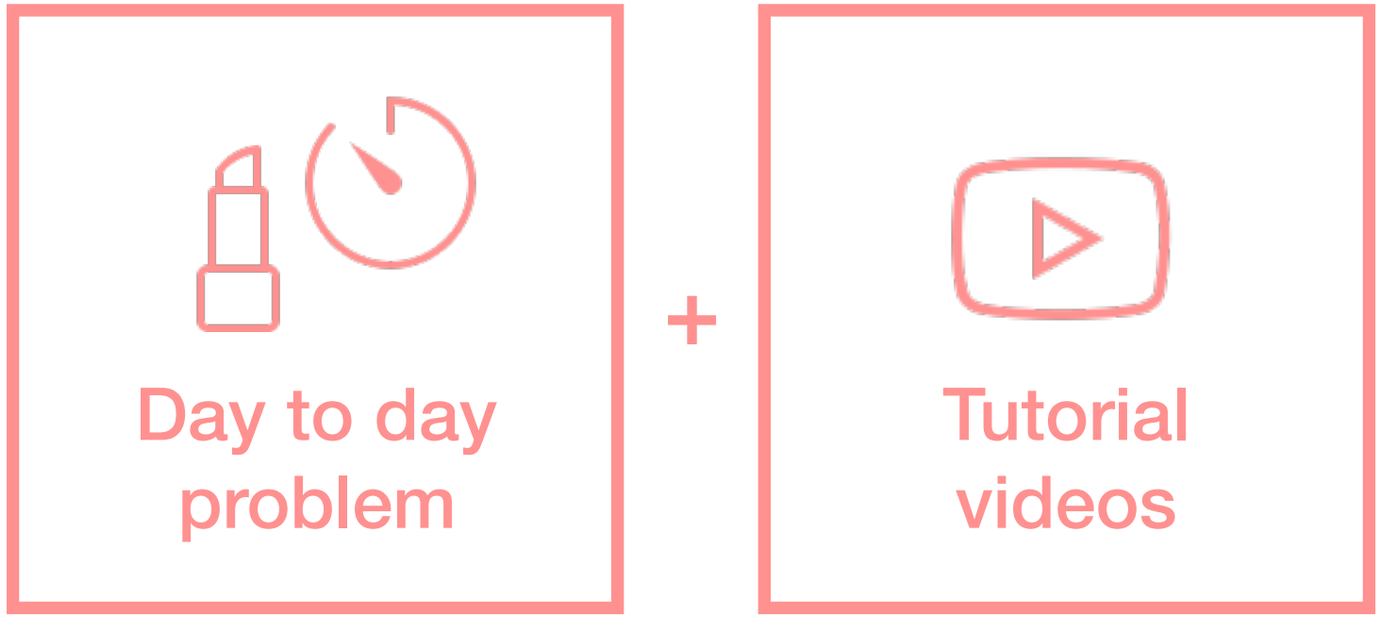
Passion

Day to day problem

Trends

1. Find a **topic**

1. Find a **topic**



1. Find a **topic**

Makeup tutorial

1. Find a **topic**

Makeup tutorial

About 32,000,000 results

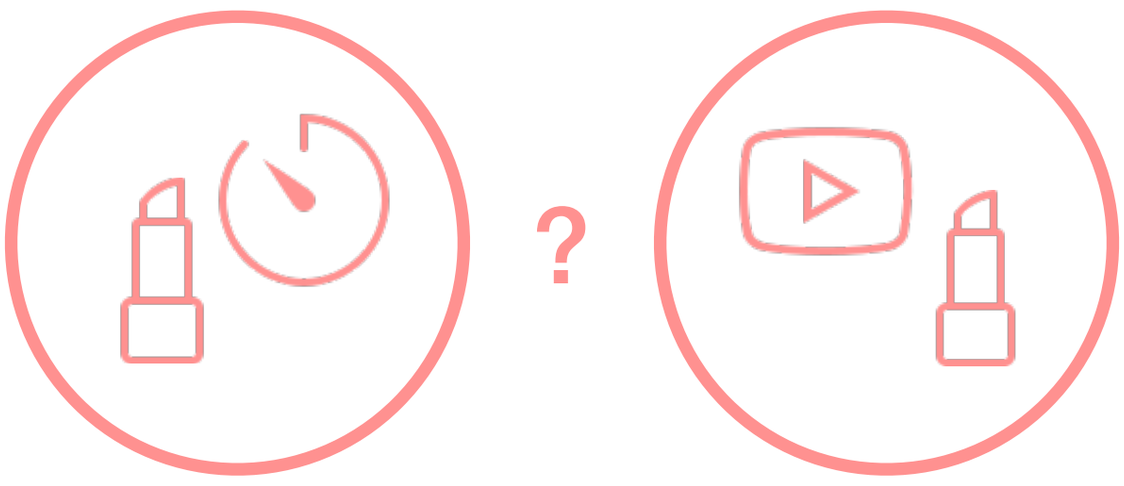
1. Find a **topic**

Makeup tutorial

About 32,000,000 results

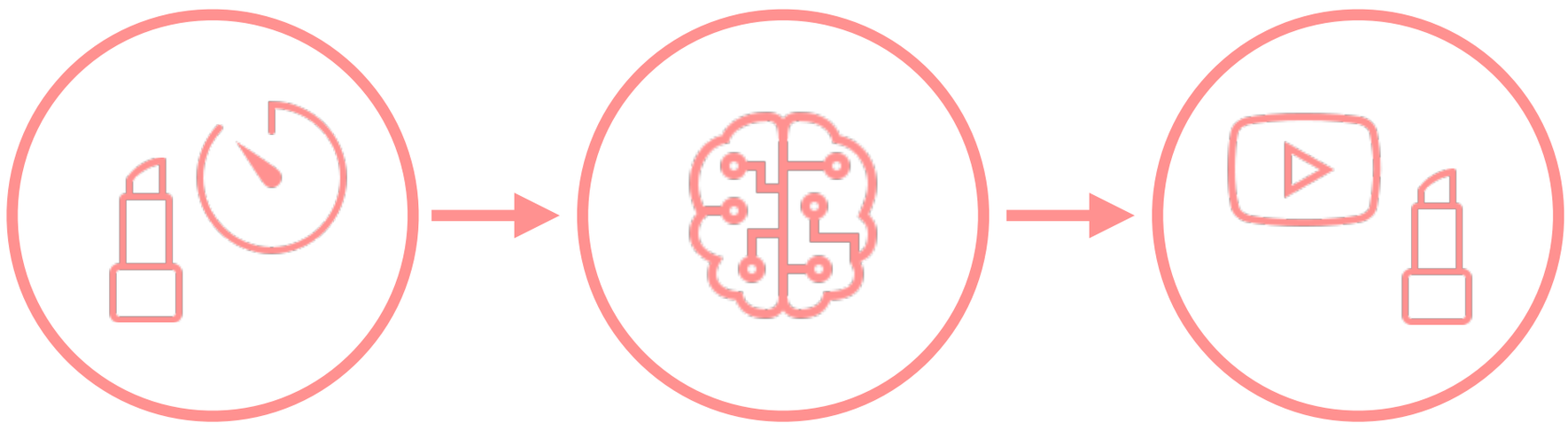
$400 billion beauty industry

1. Find a **topic**

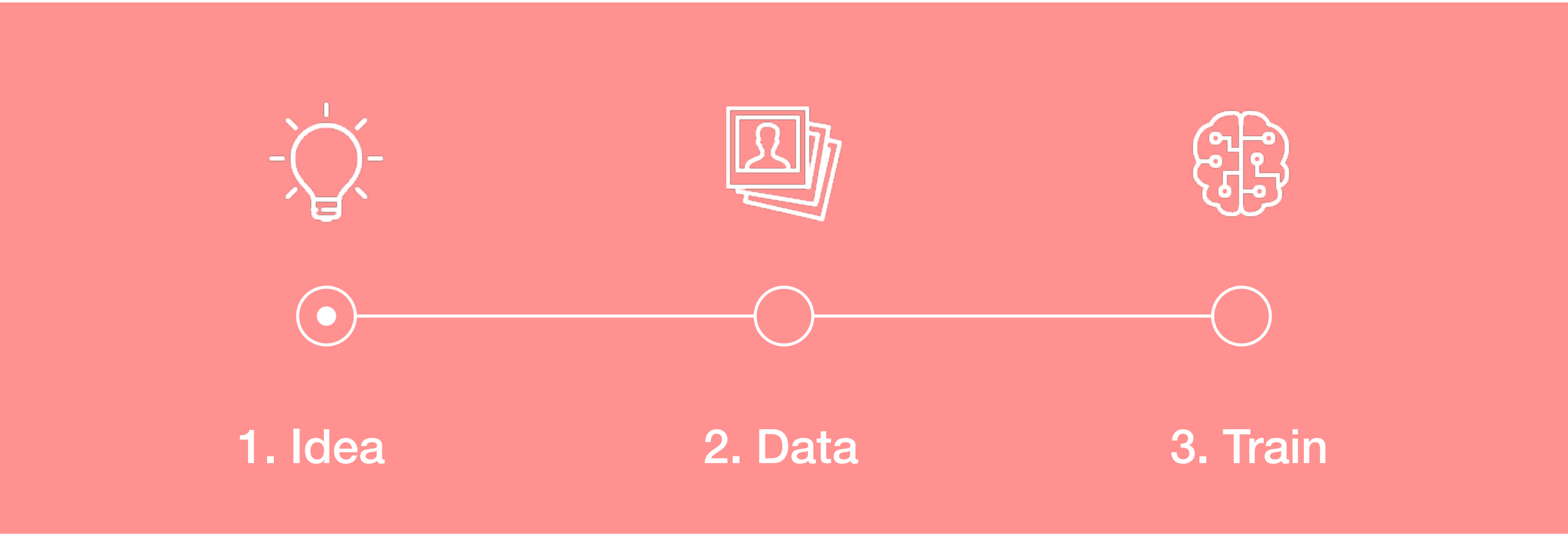


1. Find a **topic**

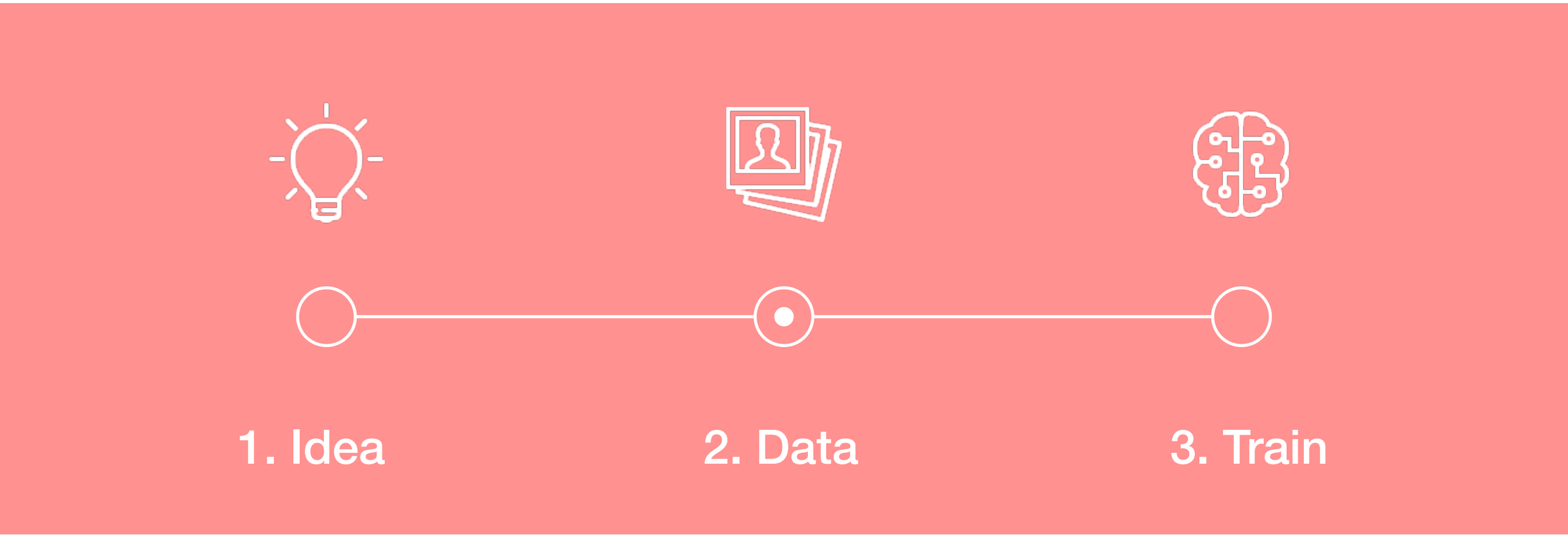
Shades



3 steps

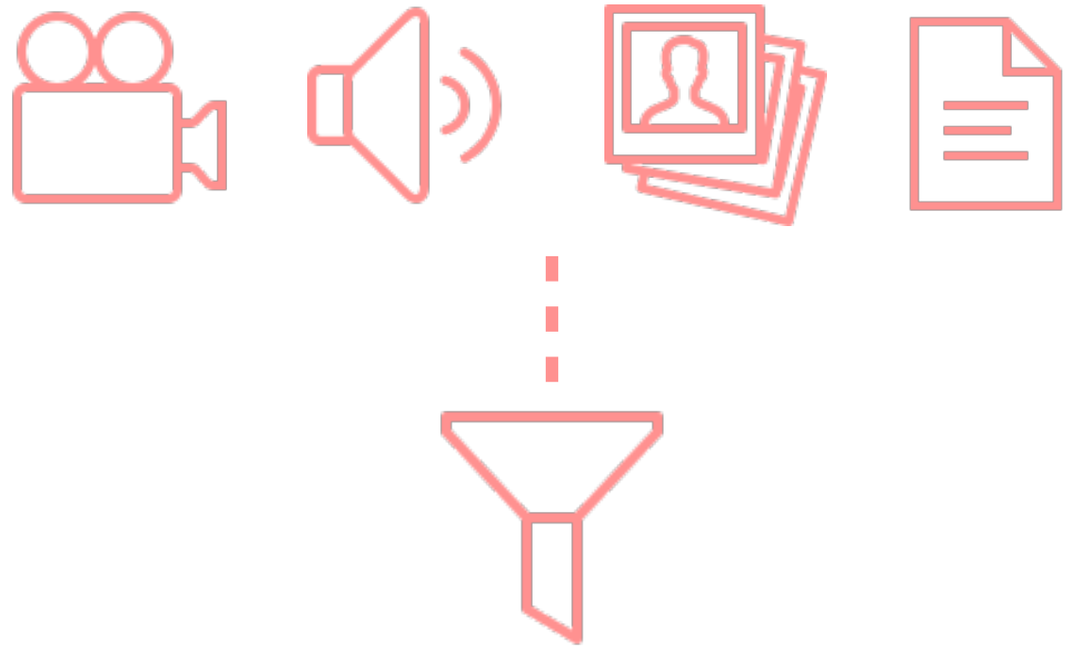


3 steps



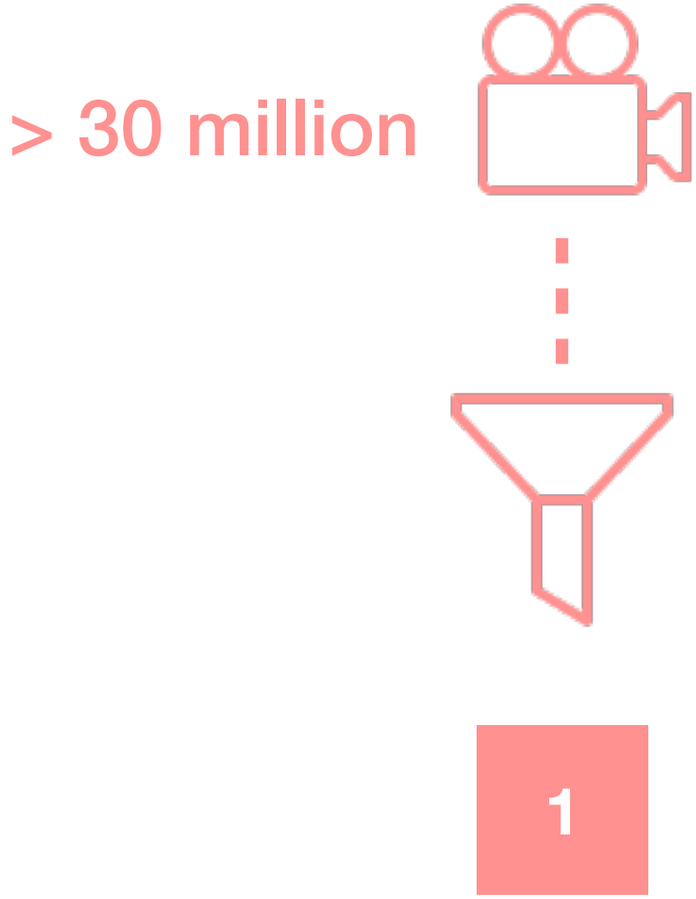
2. Create **dataset**

2. Create **dataset**



Answer

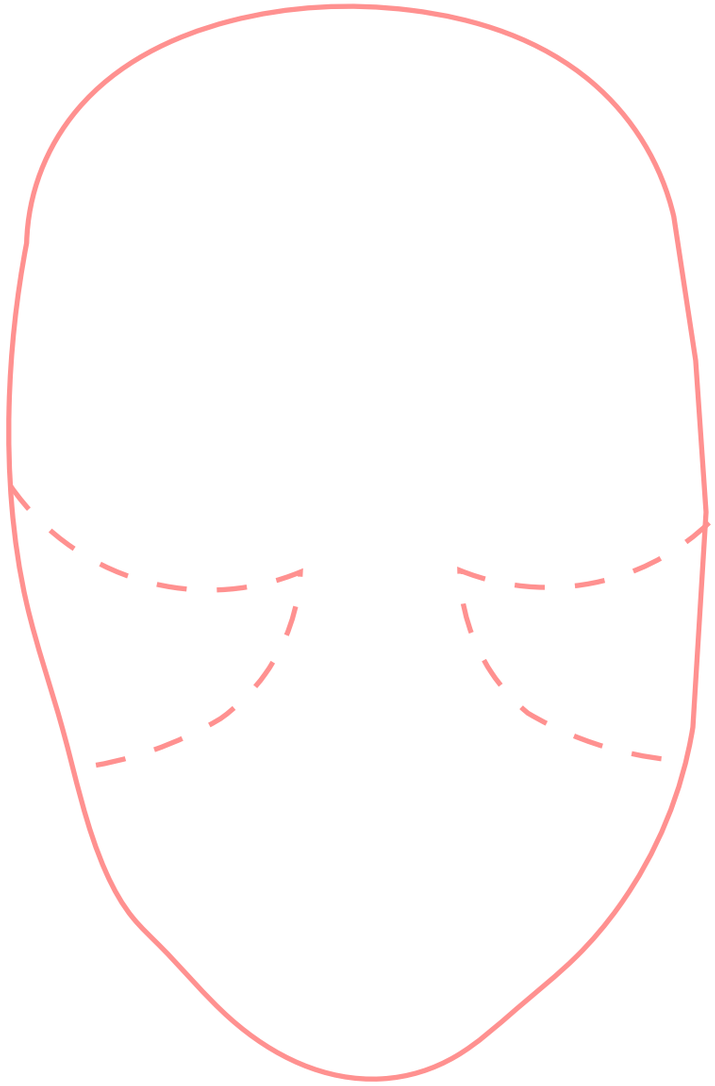
2. Create **dataset**



2. Create **dataset**

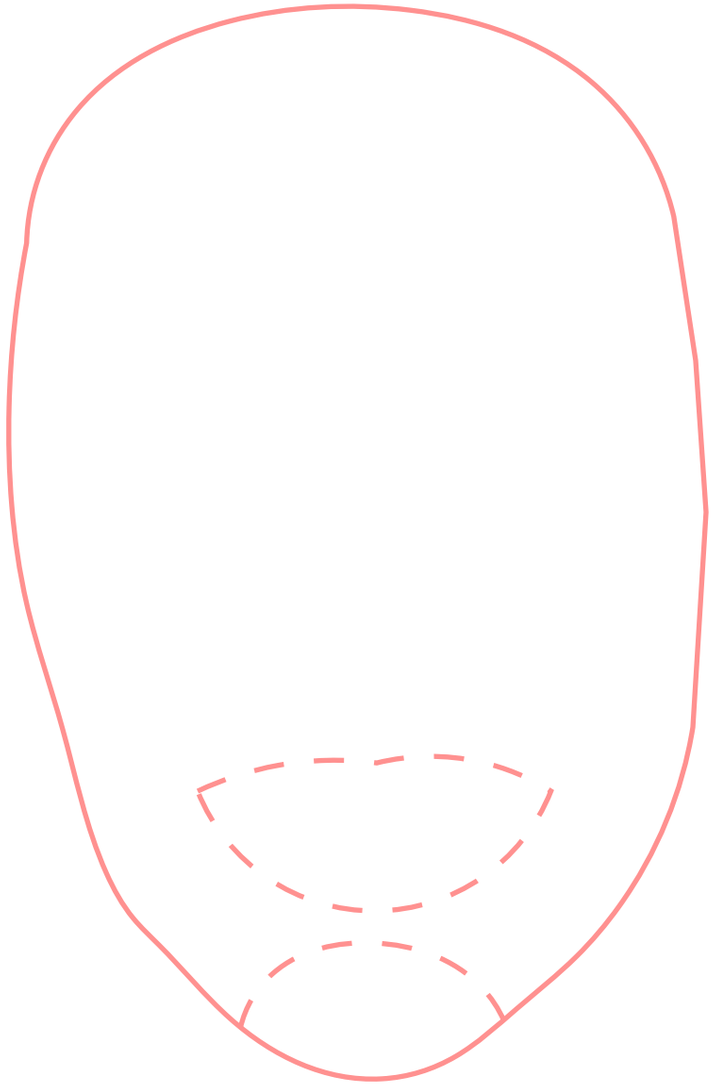
2. Create **dataset**

2. Create **dataset**

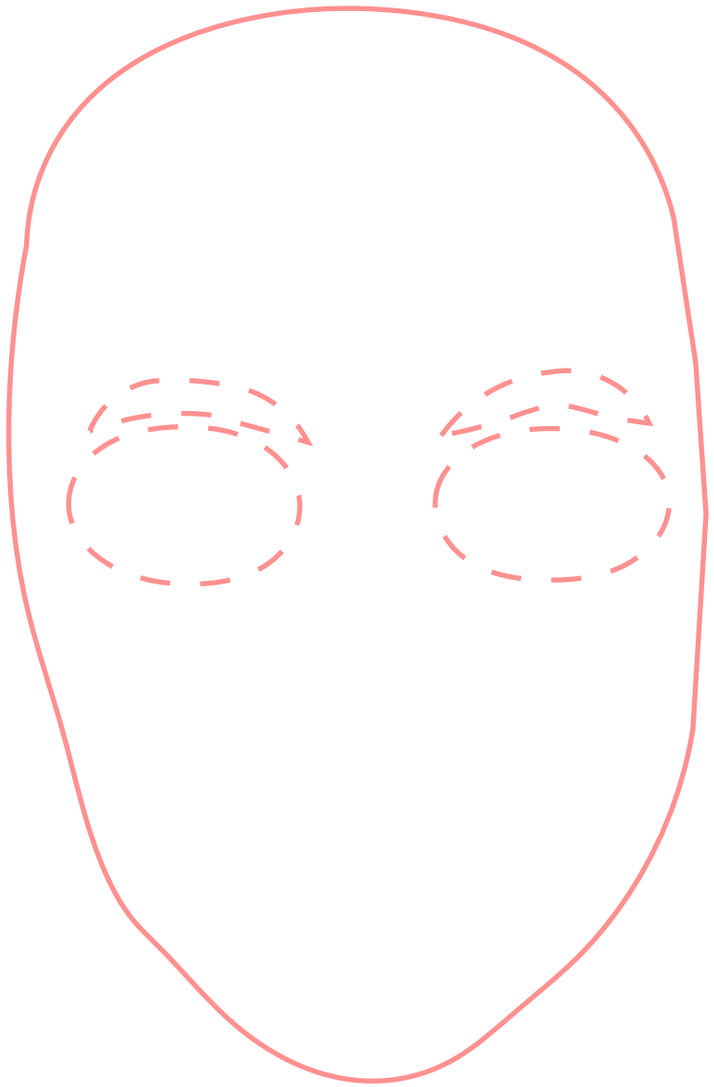


2. Create **dataset**

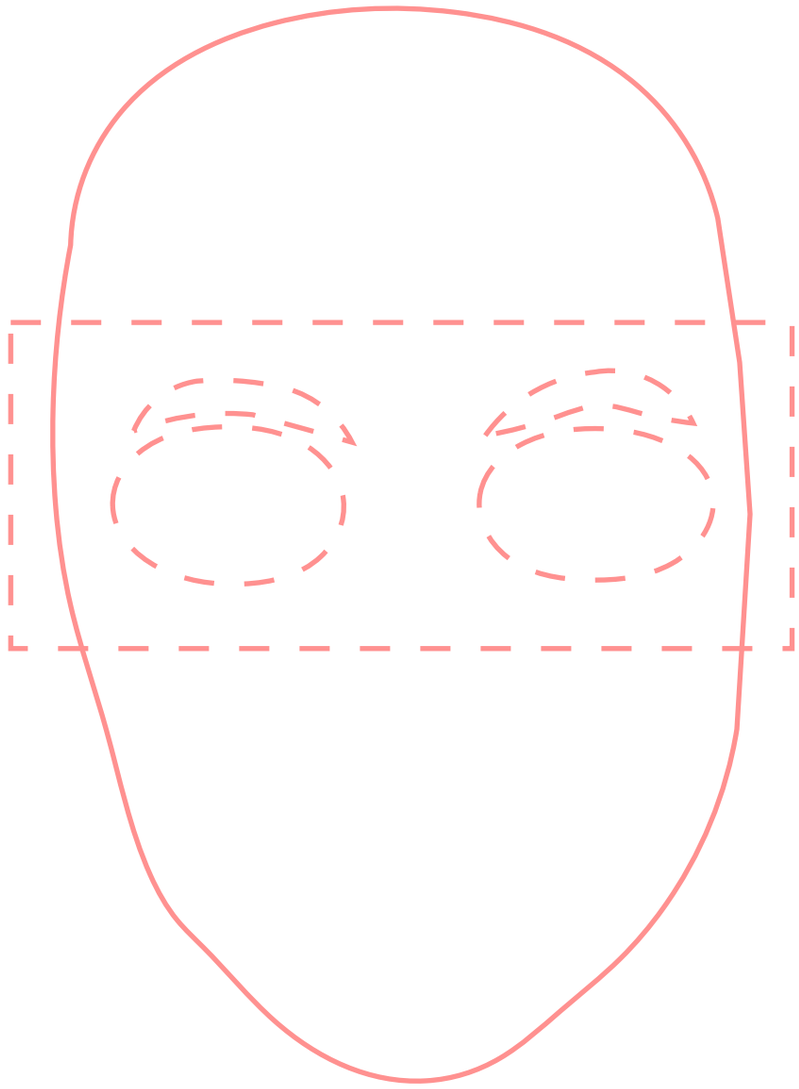
2. Create **dataset**



2. Create **dataset**



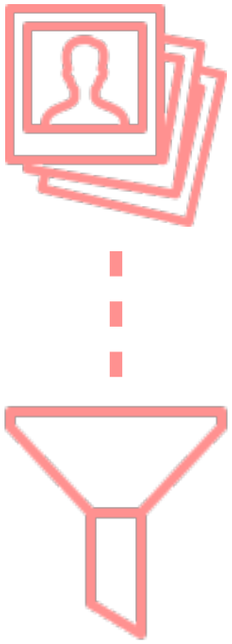
2. Create **dataset**



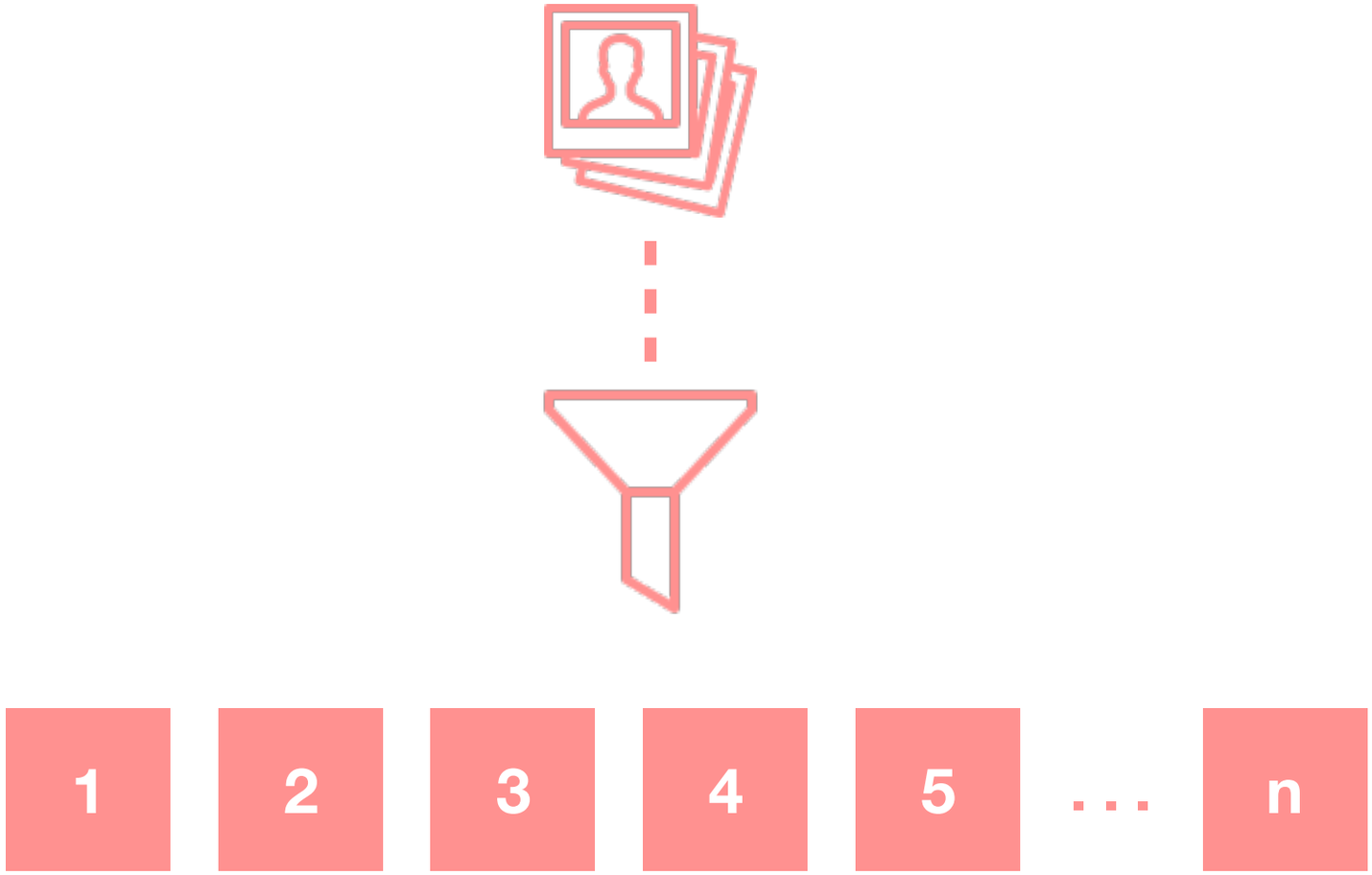
Data

Eye shapes?

2. Create **dataset**



2. Create **dataset**



2. Create **dataset**

2. Create **dataset**

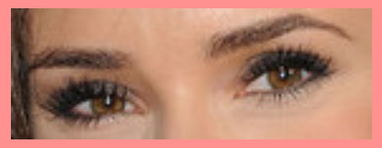
2. Create **dataset**

01 Round 02 Monolid 03 Hooded 04 Almond

2. Create **dataset**

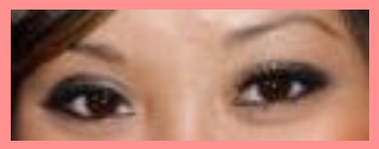
2. Create **dataset**

2. Create **dataset**



AE

HE



ME

RE

2. Create **dataset**

4 eye
categories

2. Create **dataset**

4 eye categories

200 training images

2. Create **dataset**

4 eye categories

200 training images

100 validation images

2. Create **dataset**

Training **images**

800

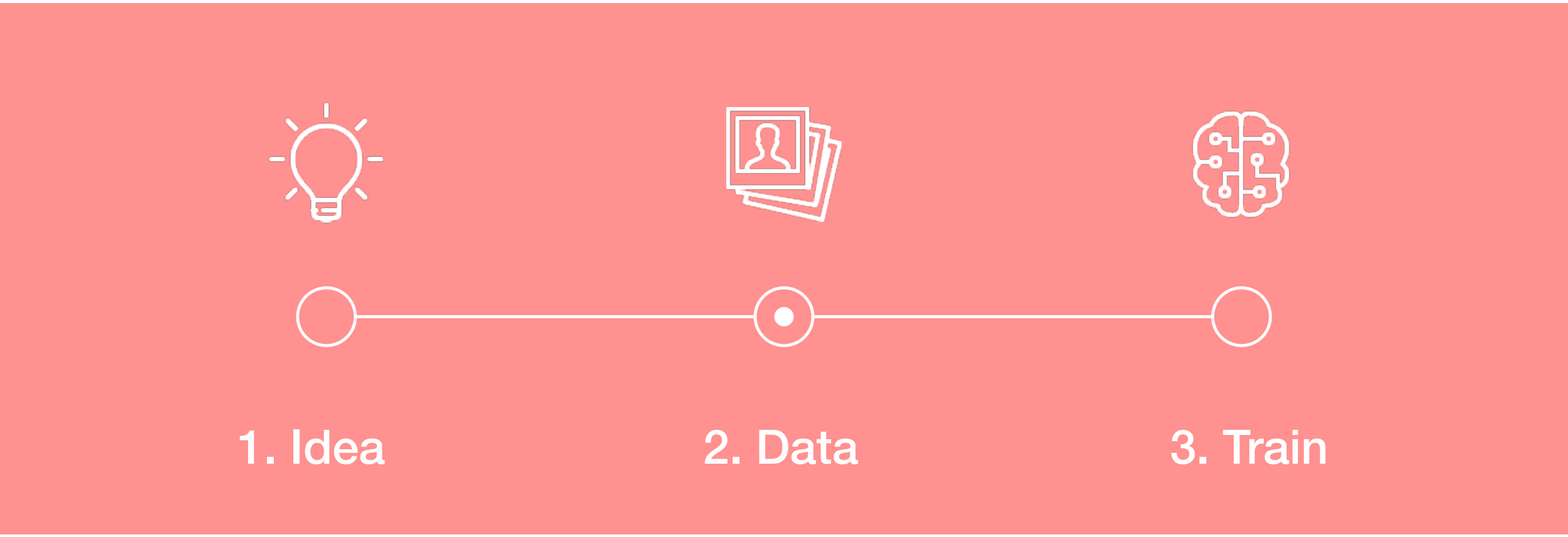
- AE
- HE
- ME
- RE

Validation **images**

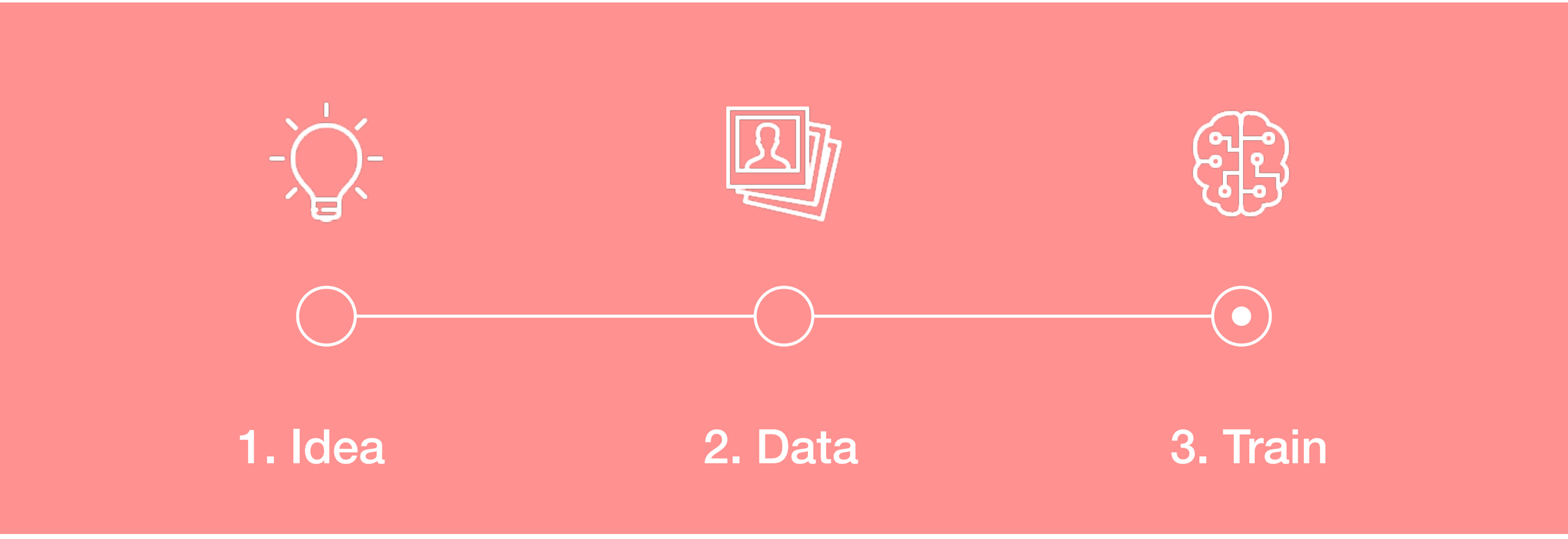
400

- AE
- HE
- ME
- RE

3 steps



3 steps



3. Train **model**

Learning from scratch

Conv2D
Conv2D
MaxP2D

Conv2D
Conv2D
MaxP2D

3. Train **model**

Learning from scratch

Conv2D
Conv2D
MaxP2D

Conv2D
Conv2D
MaxP2D

Transfer learning

Conv2D
Conv2D
MaxP2D

Conv2D
Conv2D
MaxP2D

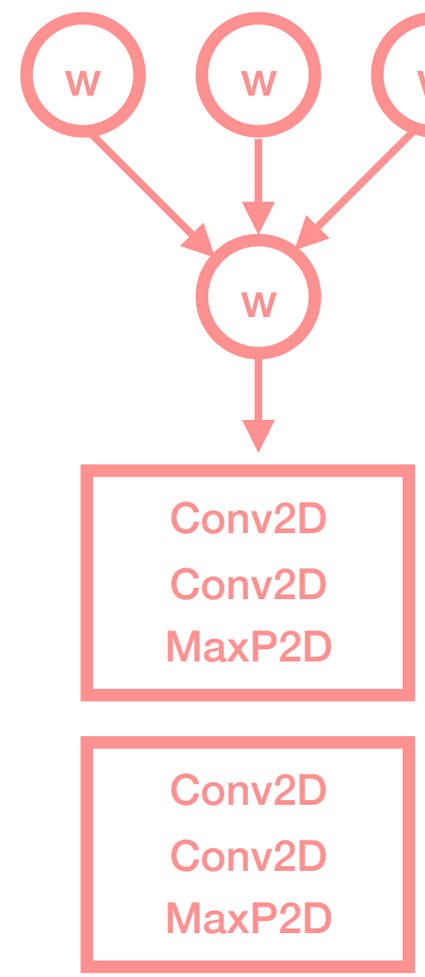
3. Train **model**

Learning from scratch

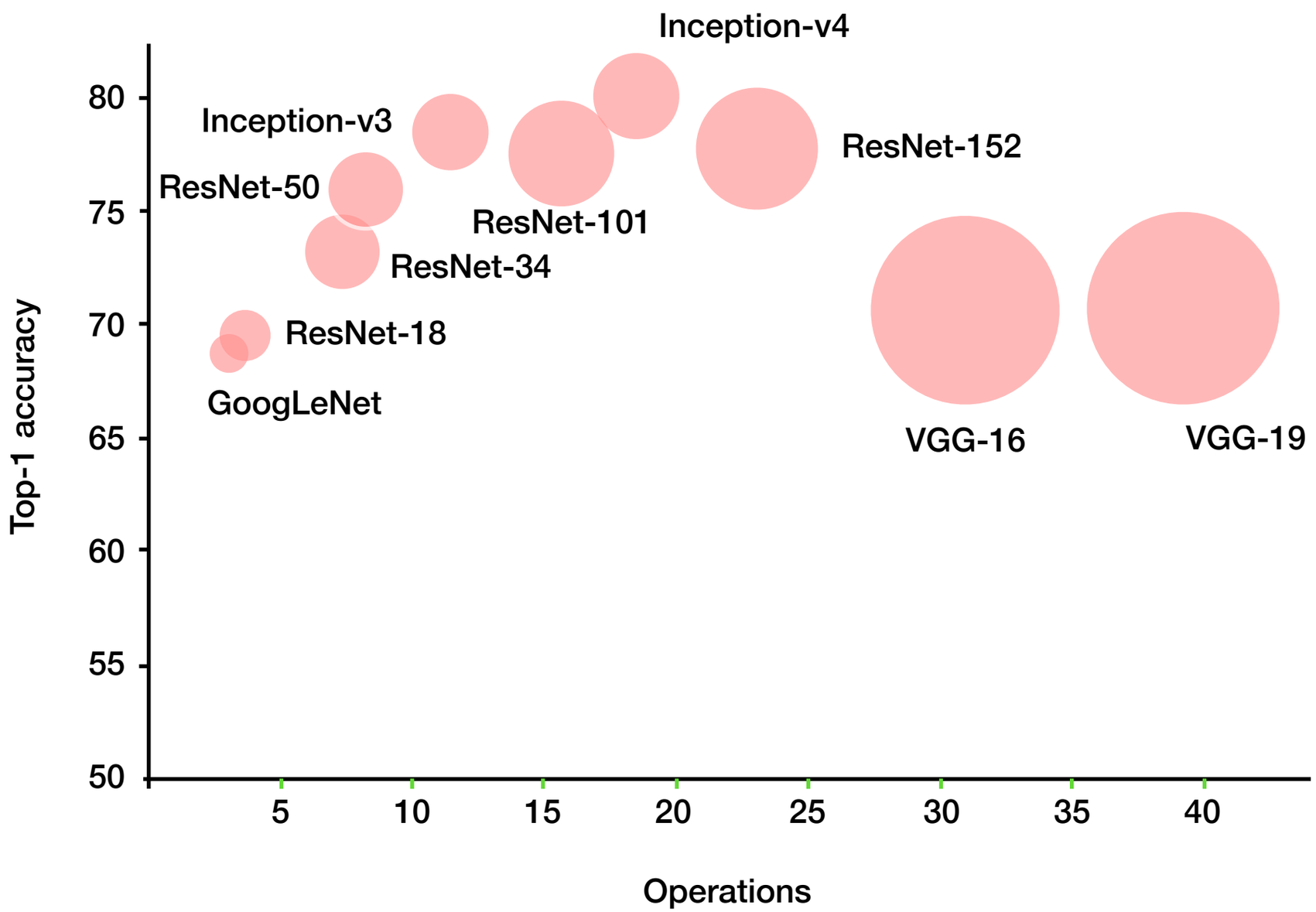
Conv2D
Conv2D
MaxP2D

Conv2D
Conv2D
MaxP2D

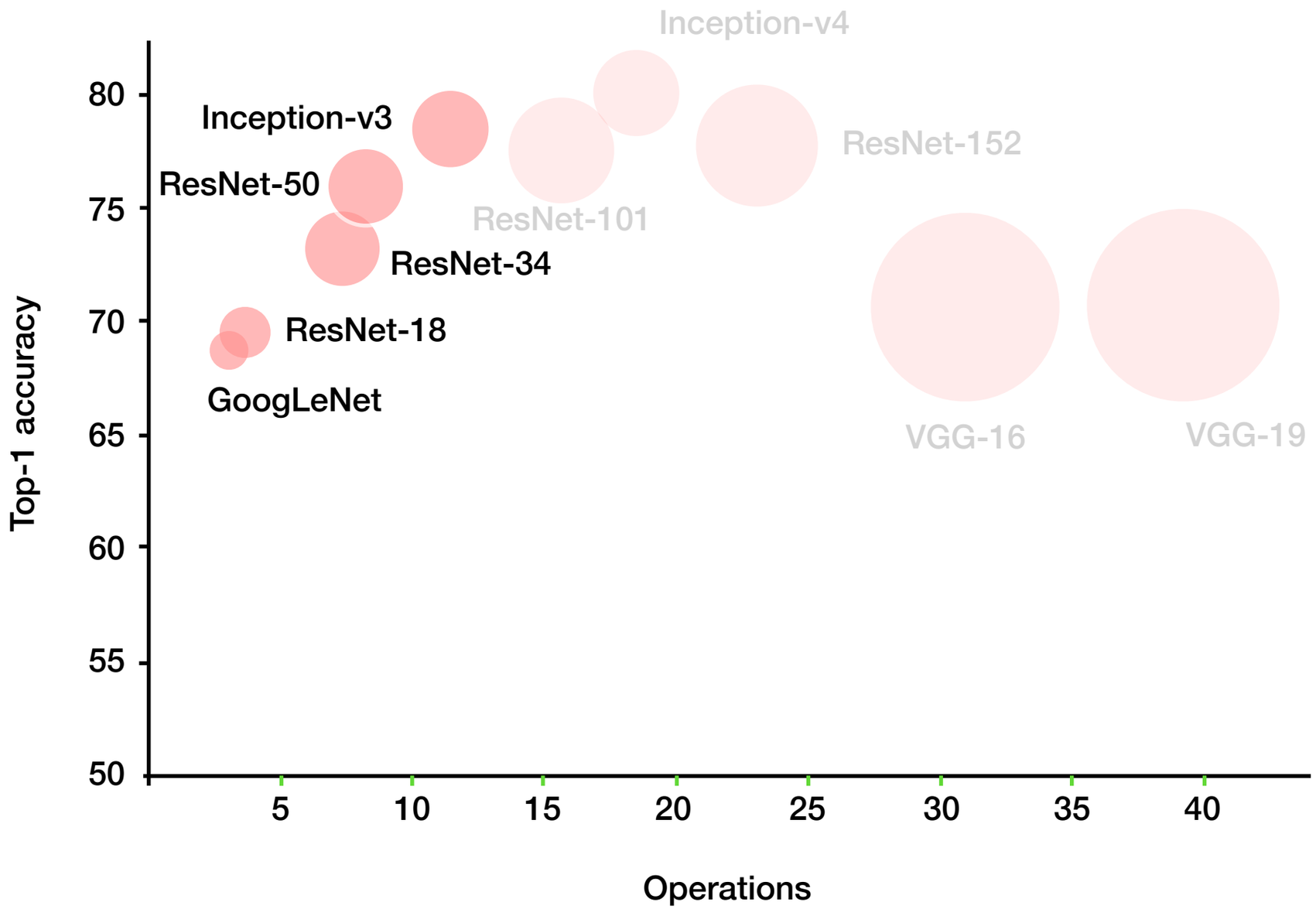
Transfer learning



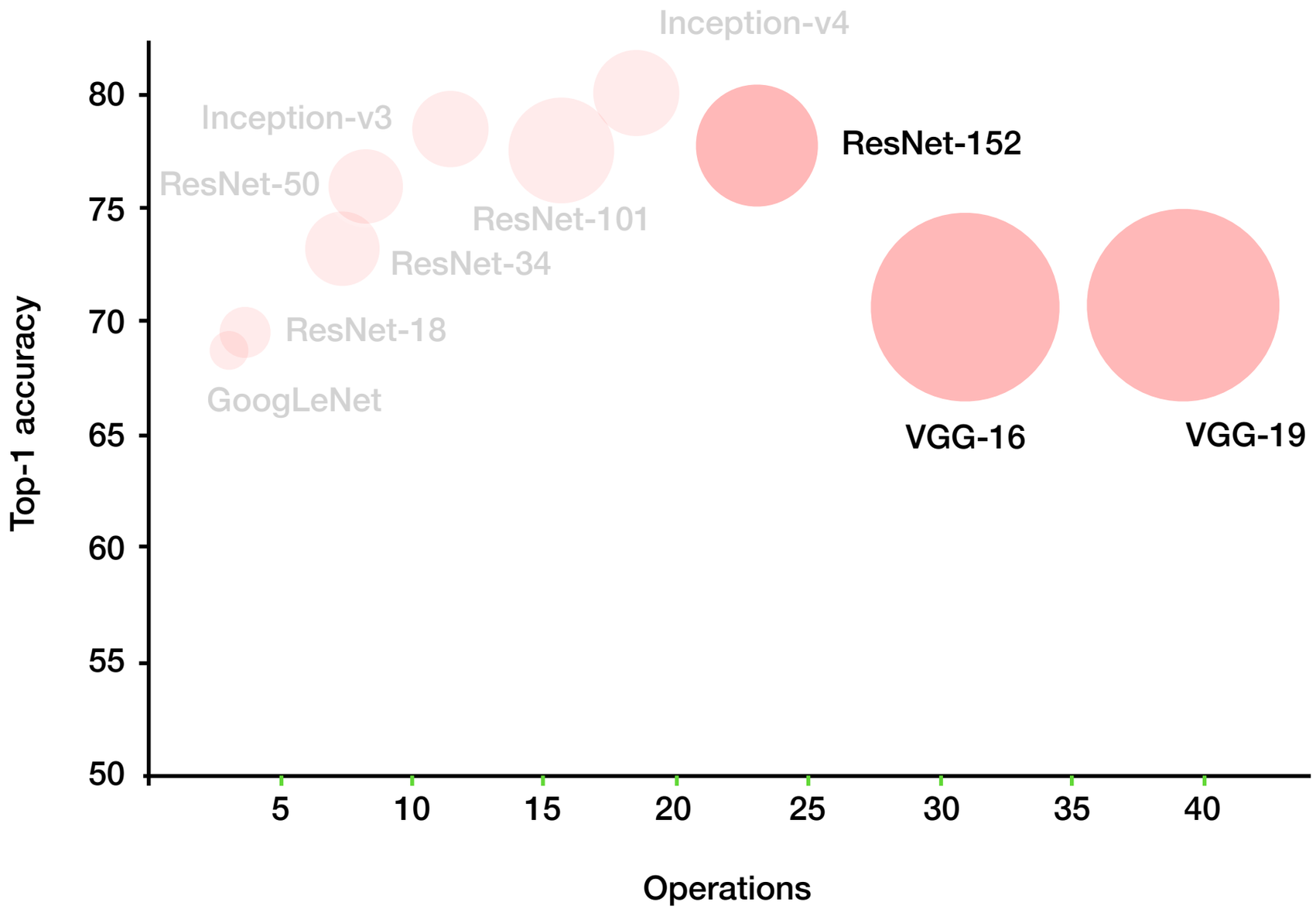
3. Train **model**



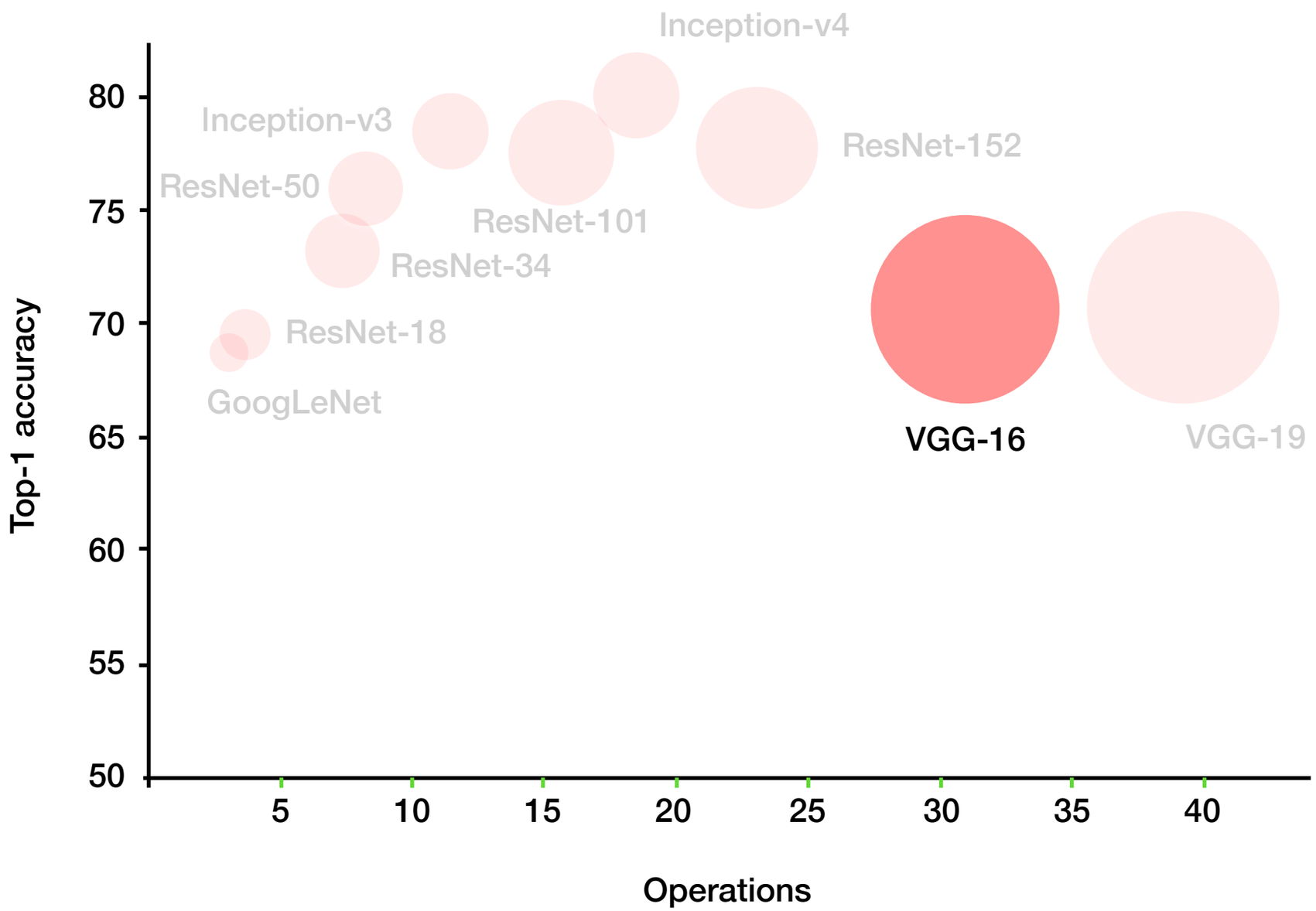
3. Train **model**



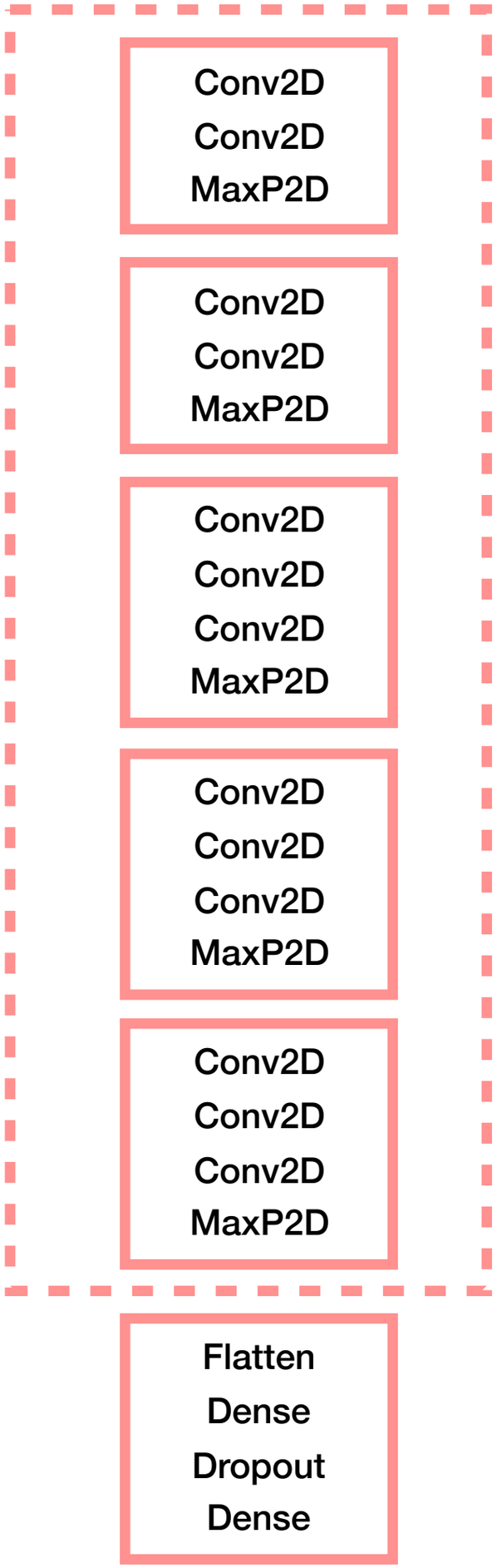
3. Train **model**



3. Train **model**

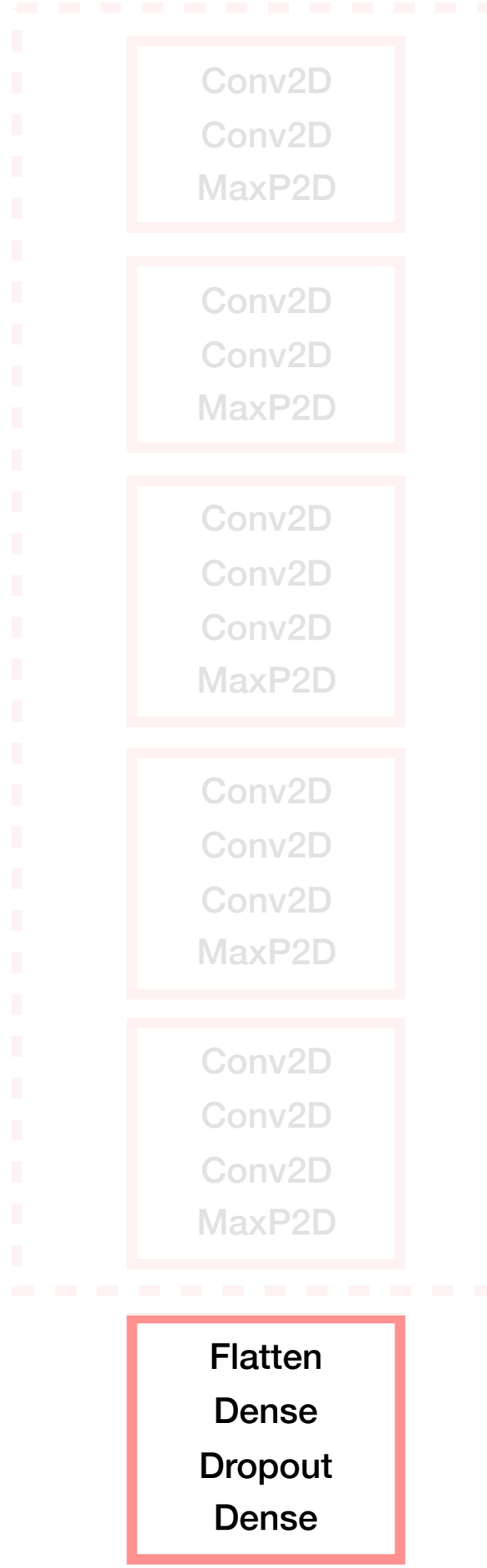


3. Train **model**



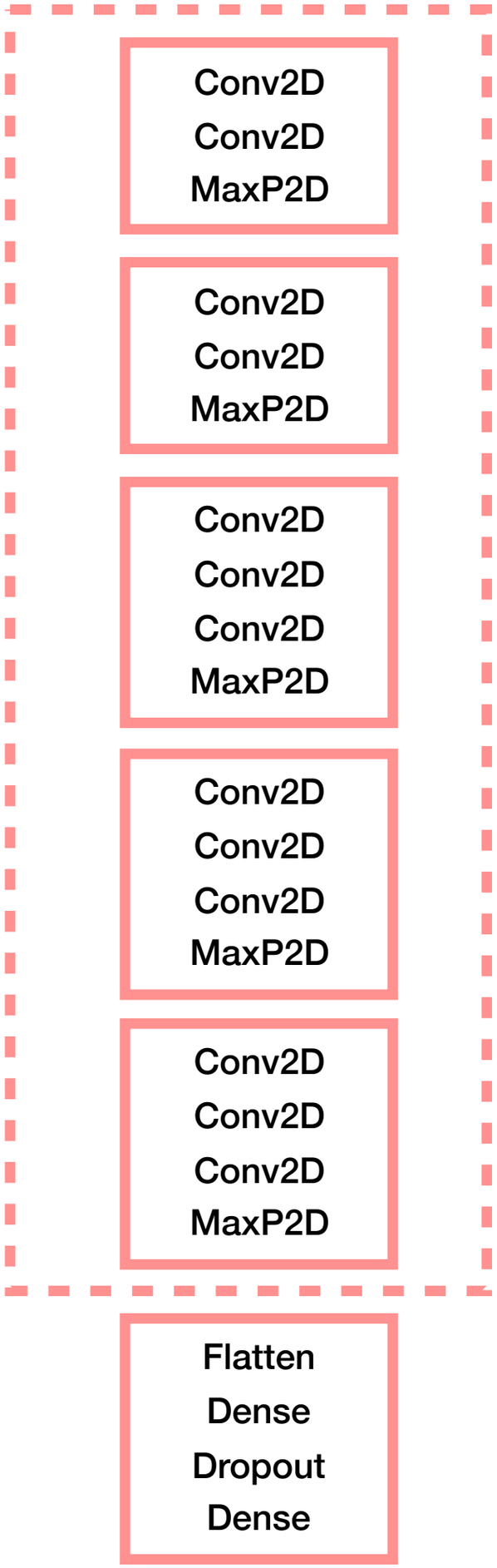
Norah Klintberg Sakal

3. Train **model**



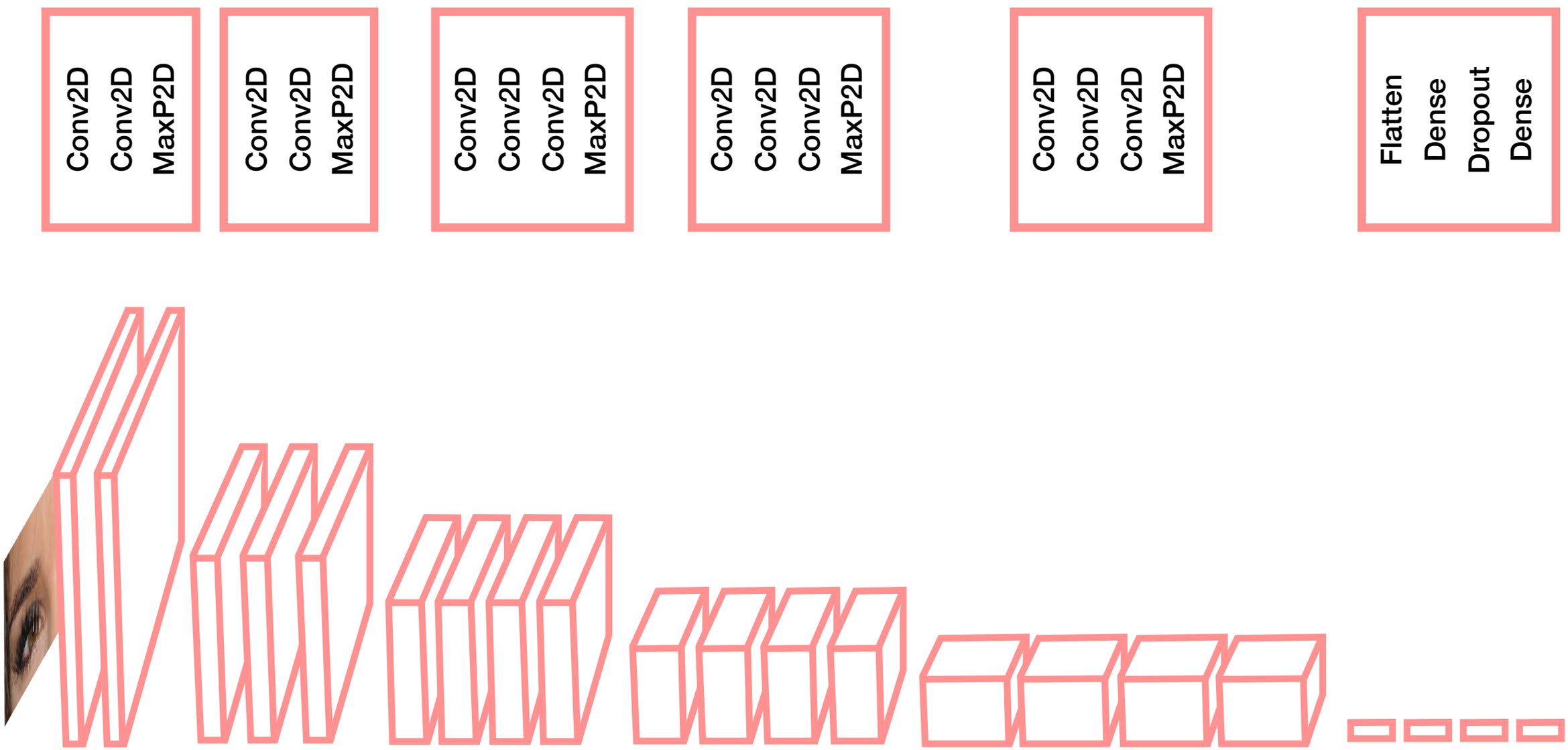
Norah Klintberg Sakal

3. Train **model**

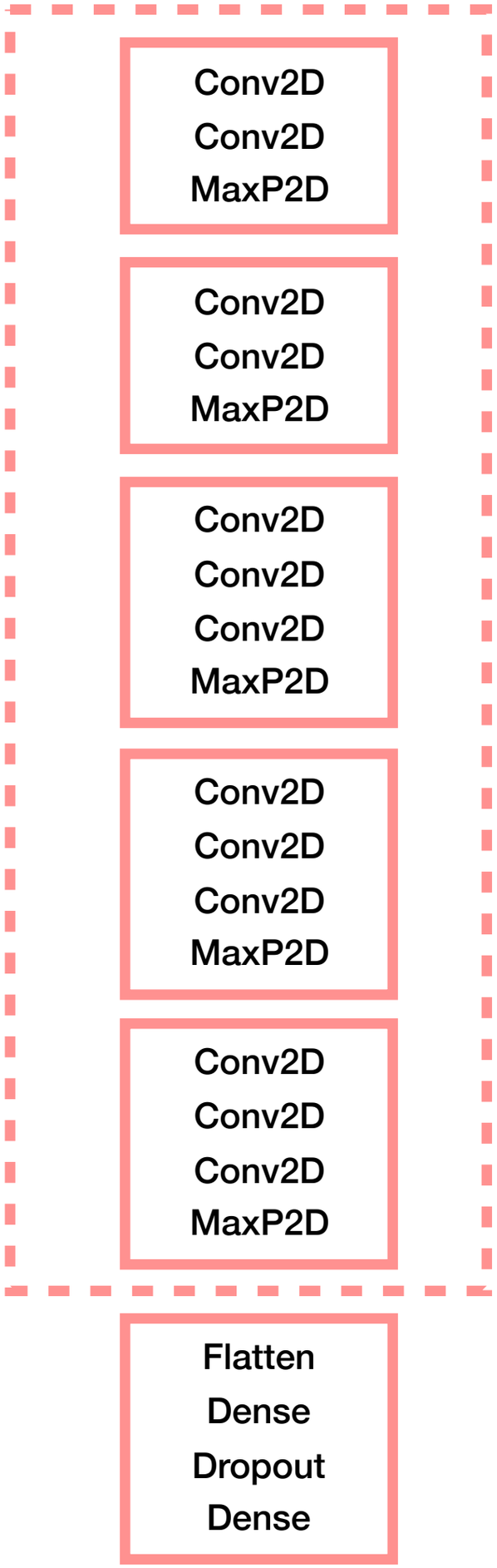


Norah Klintberg Sakal

3. Train **model**



3. Train **model**



Norah Klintberg Sakal

3. Train **model**

Conv2D
Conv2D
MaxP2D

3. Train **model**

Conv2D

Conv2D

MaxP2D

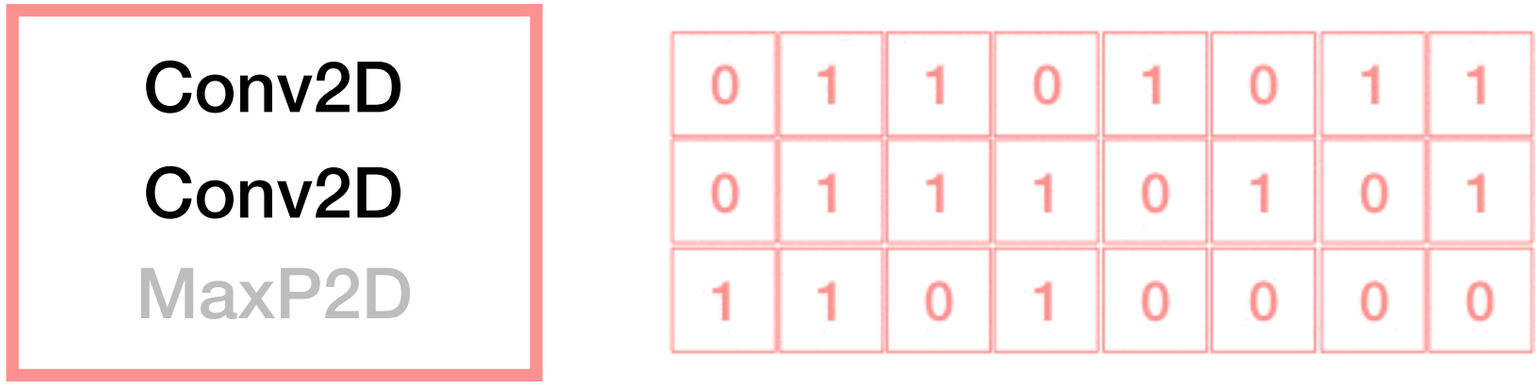
3. Train **model**

Conv2D

Conv2D

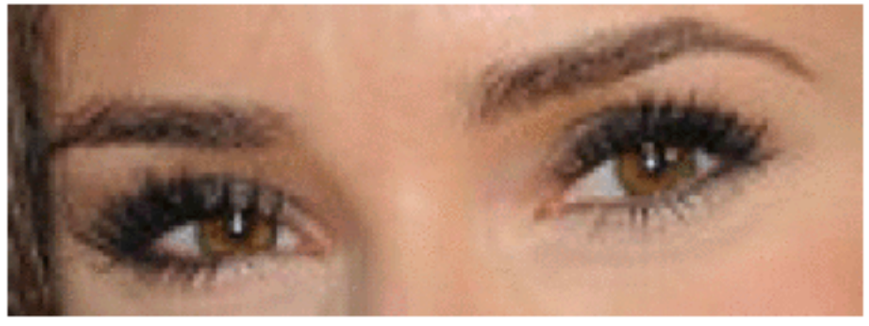
MaxP2D

3. Train **model**



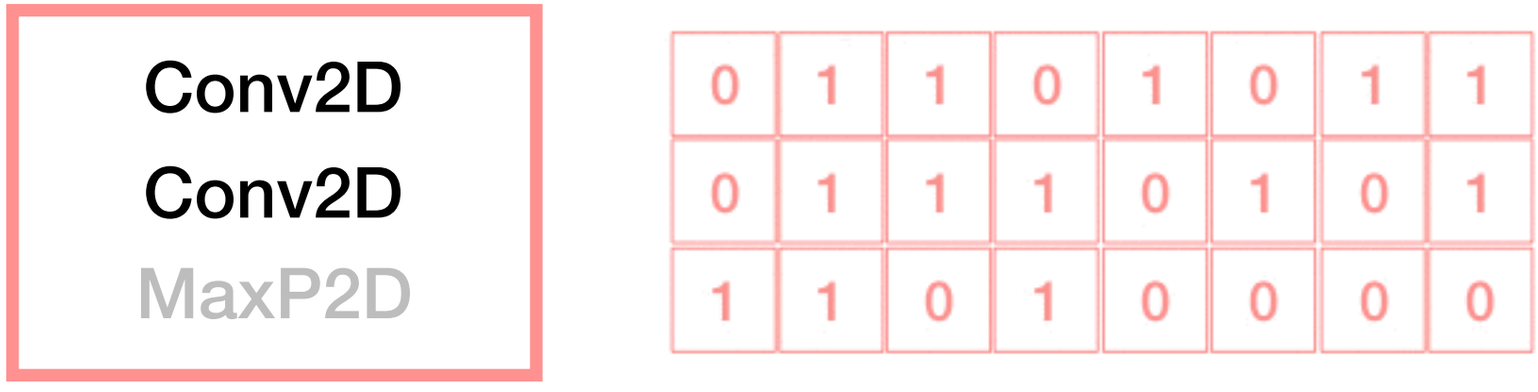
3. Train **model**

Conv2D
Conv2D
MaxP2D



Norah Klintberg Sakal

3. Train **model**

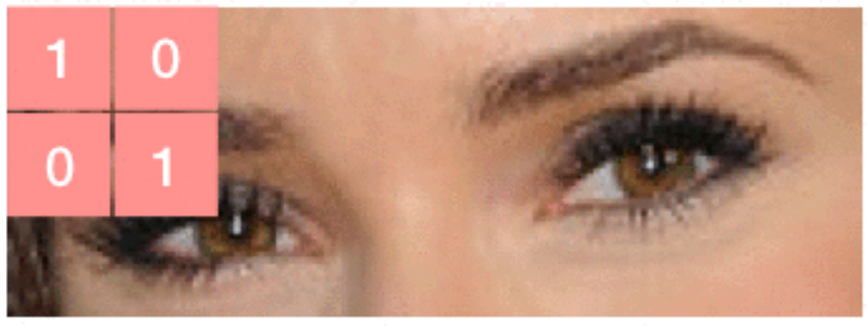


3. Train **model**

Conv2D

Conv2D

MaxP2D



1

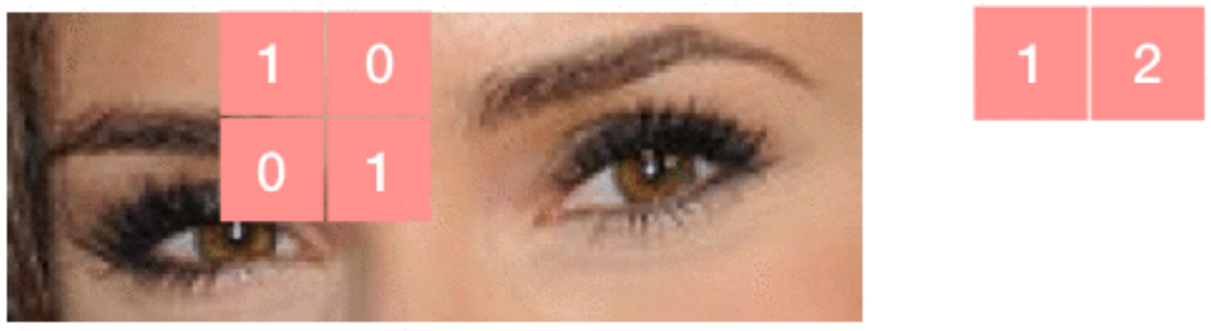
Norah Klintberg Sakal

3. Train **model**

Conv2D

Conv2D

MaxP2D



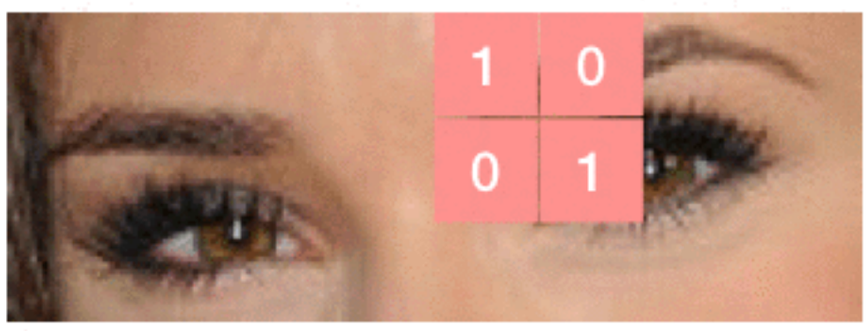
Norah Klintberg Sakal

3. Train **model**

Conv2D

Conv2D

MaxP2D



1	2	2

Norah Klintberg Sakal

3. Train **model**

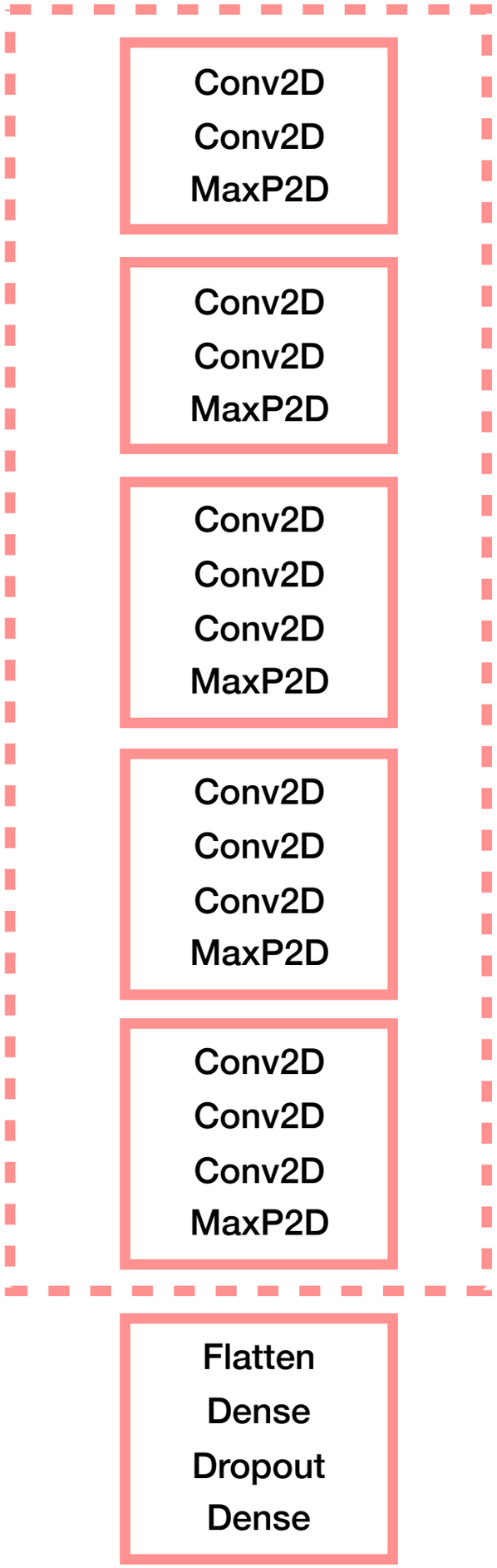
Conv2D
Conv2D
MaxP2D

1	2	2	2
1	2	0	0

Feature **map**

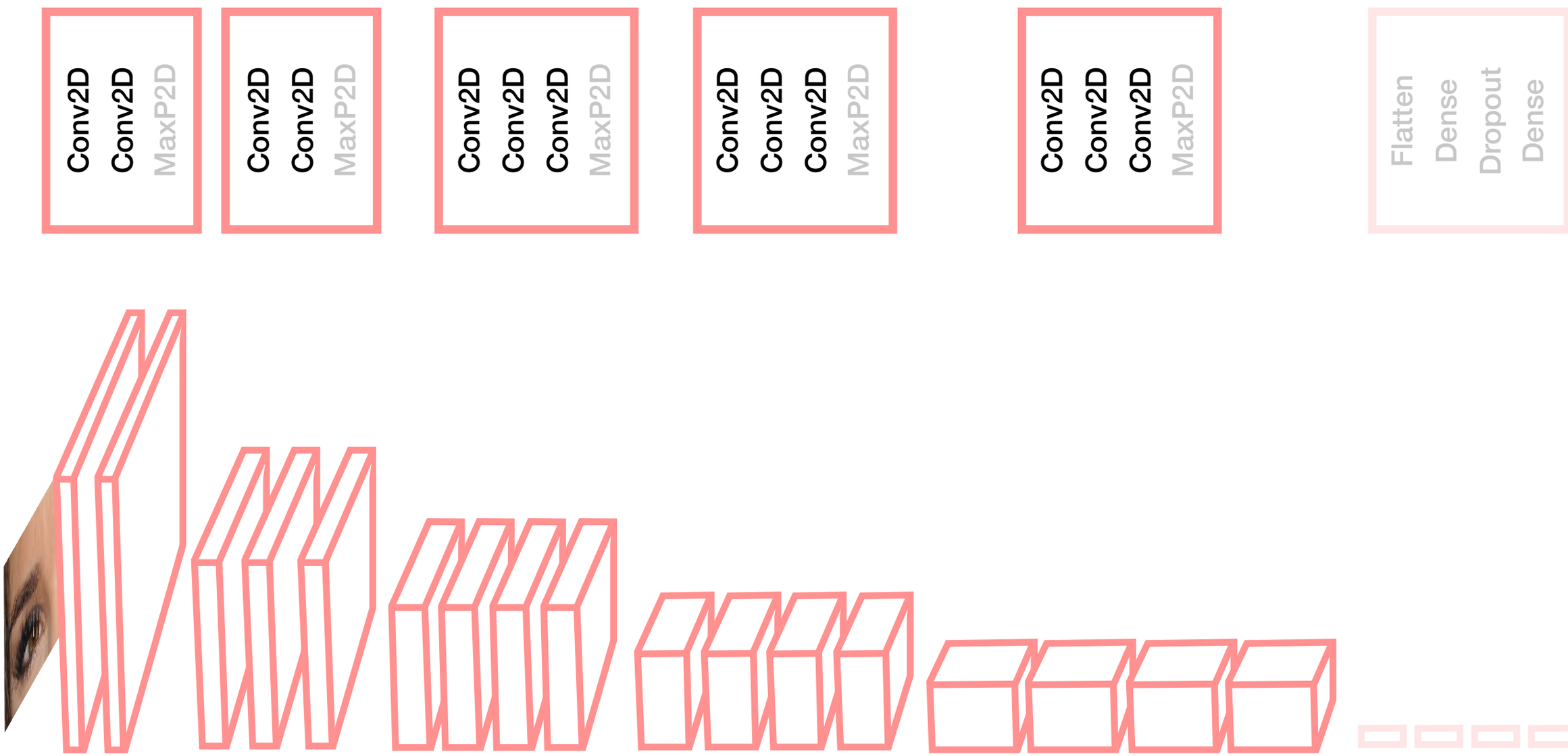
Norah Klintberg Sakal

3. Train **model**



Norah Klintberg Sakal

3. Train **model**



3. Train **model**

Conv2D

Conv2D

MaxP2D

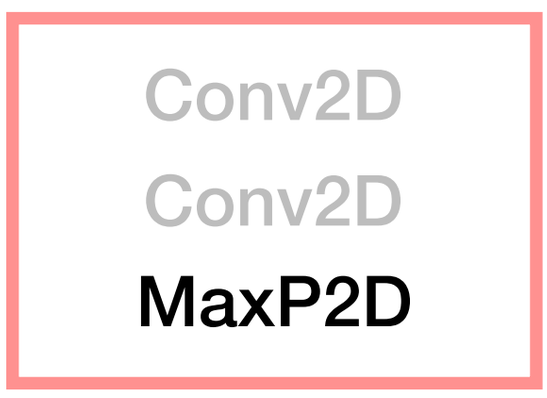
Pooling **step**

1	2	2	2
1	2	0	0

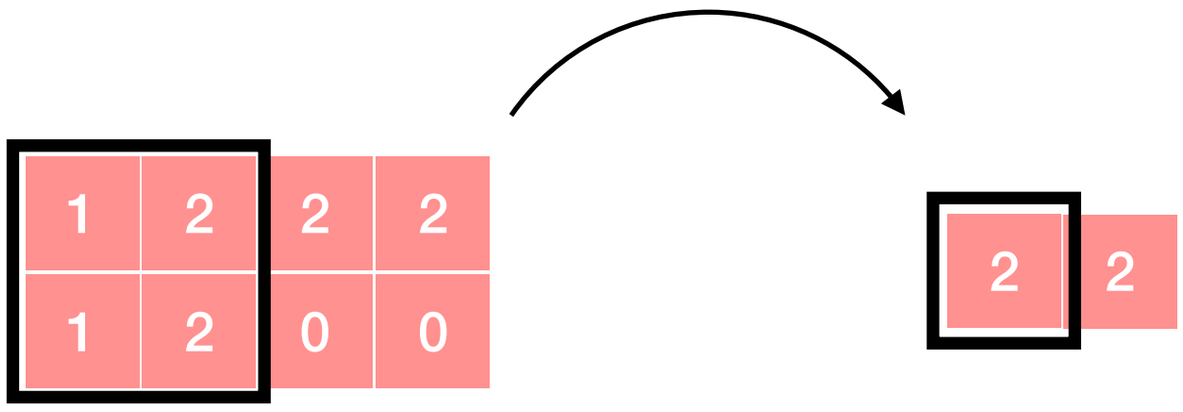
Feature **map**

2	2

3. Train **model**



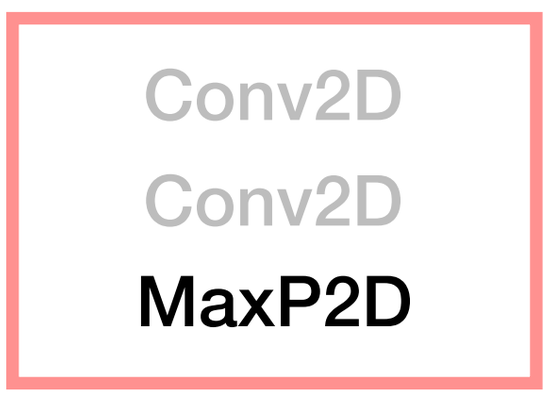
Pooling **step**

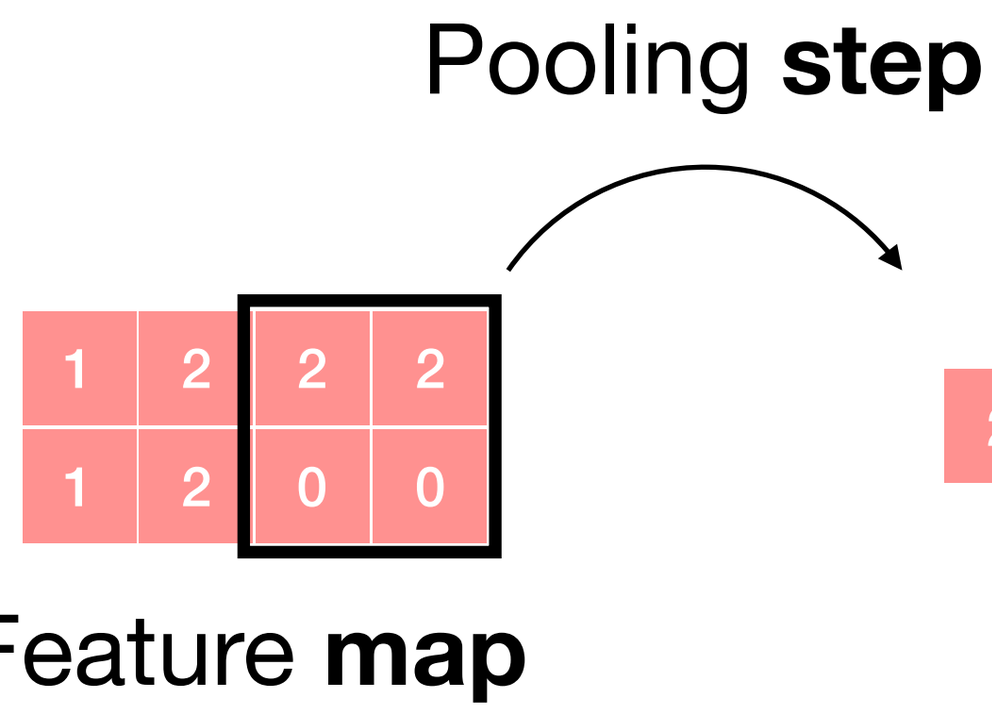


Feature **map**

Norah Klintberg Sakal

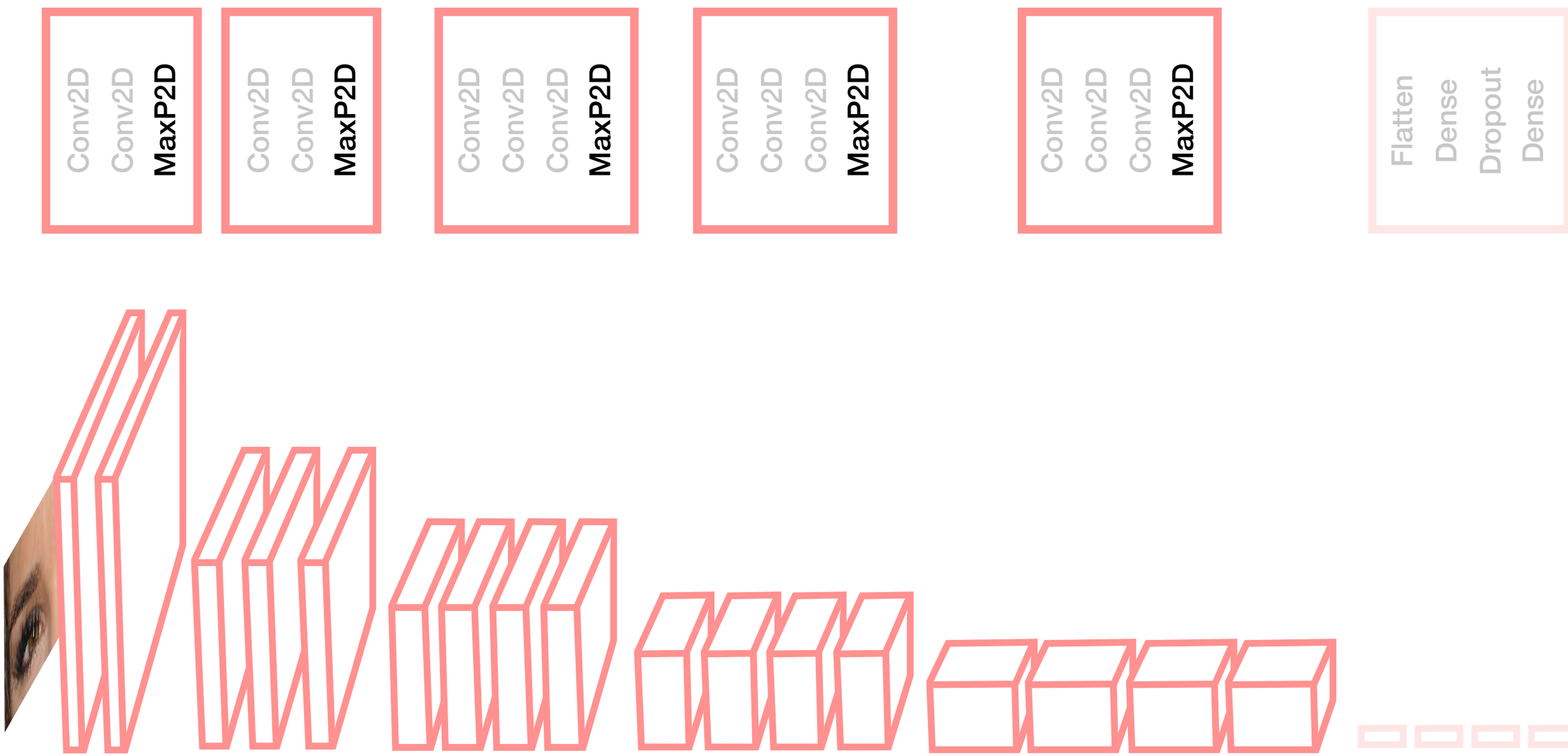
3. Train **model**



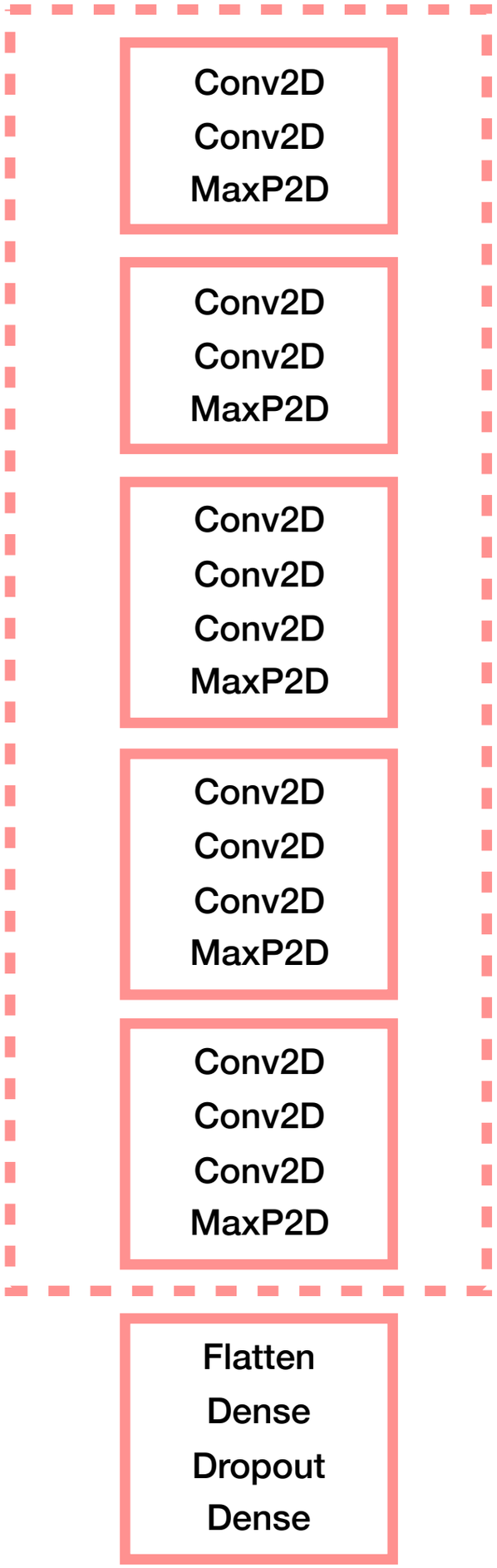


Norah Klintberg Sakal

3. Train **model**

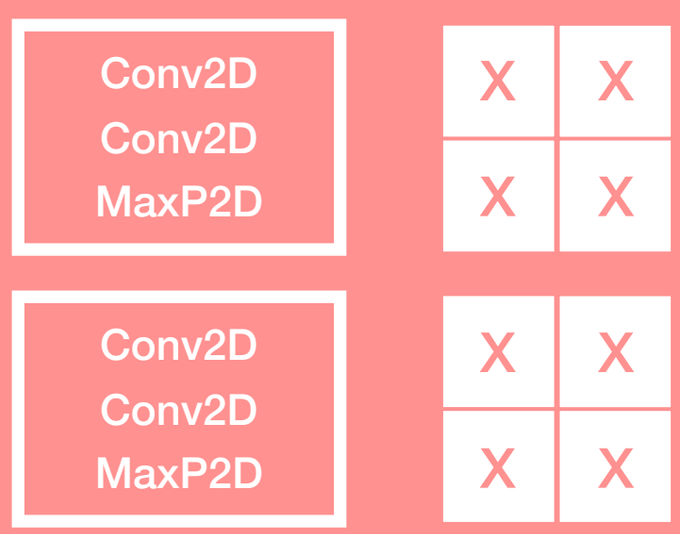


3. Train **model**

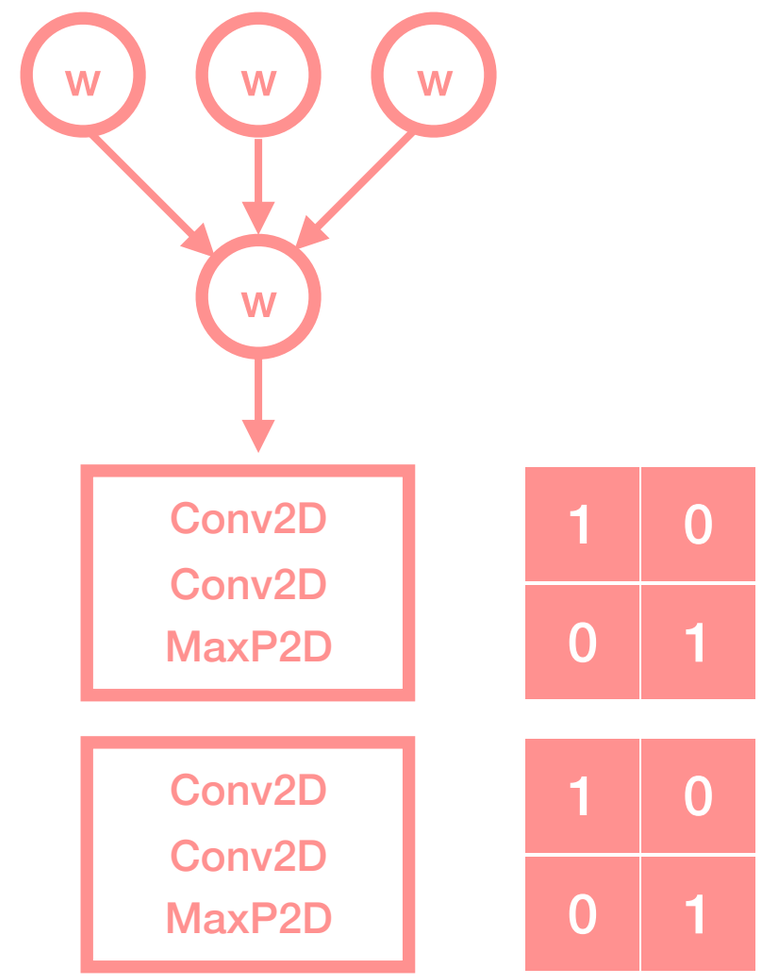


Norah Klintberg Sakal

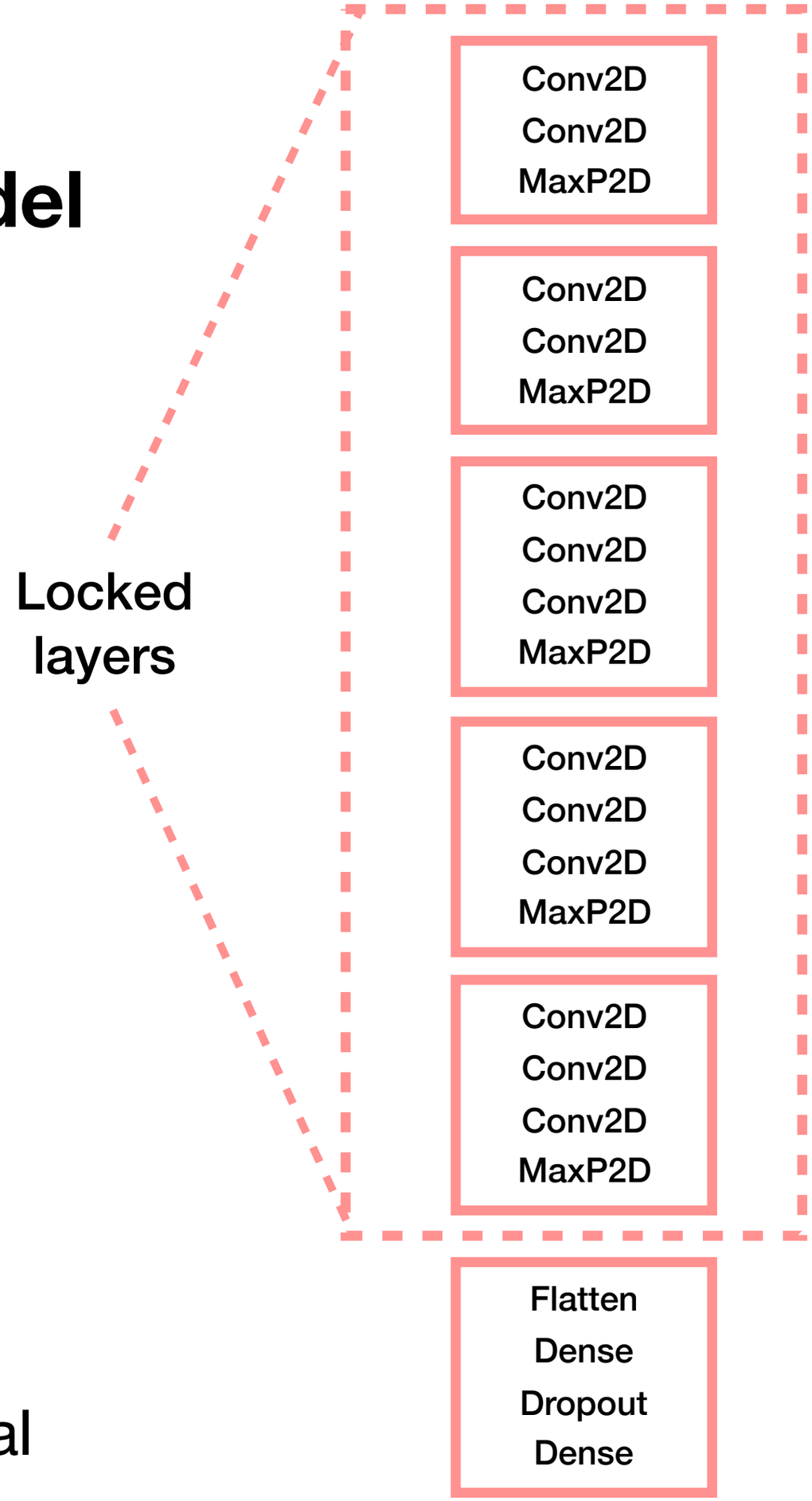
Learning from scratch



Transfer learning



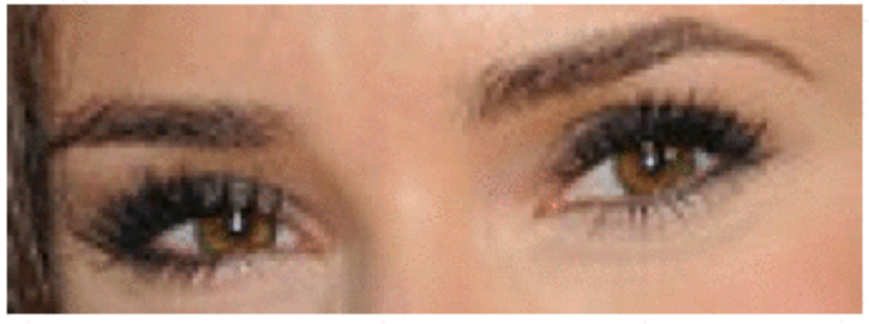
3. Train **model**



Norah Klintberg Sakal

3. Train **model**

Conv2D
Conv2D
MaxP2D



1	0
0	1

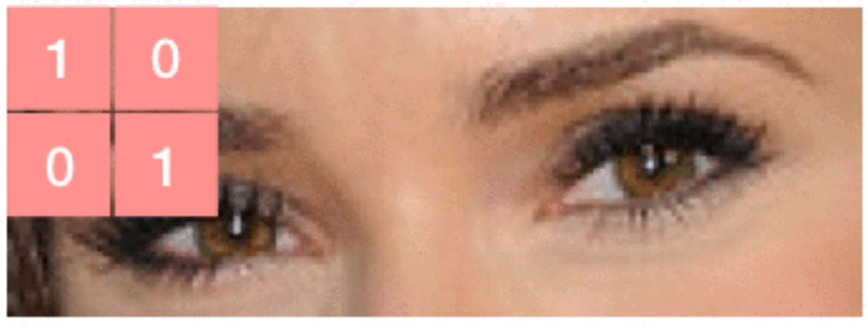
Norah Klintberg Sakal

3. Train **model**

Conv2D

Conv2D

MaxP2D

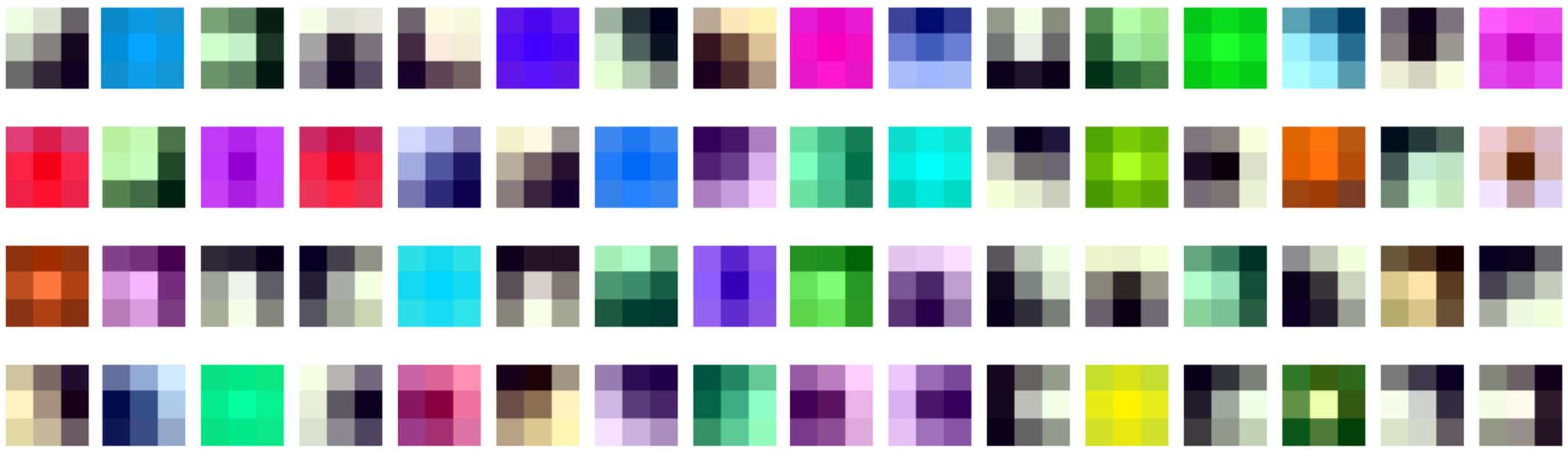


1

Norah Klintberg Sakal

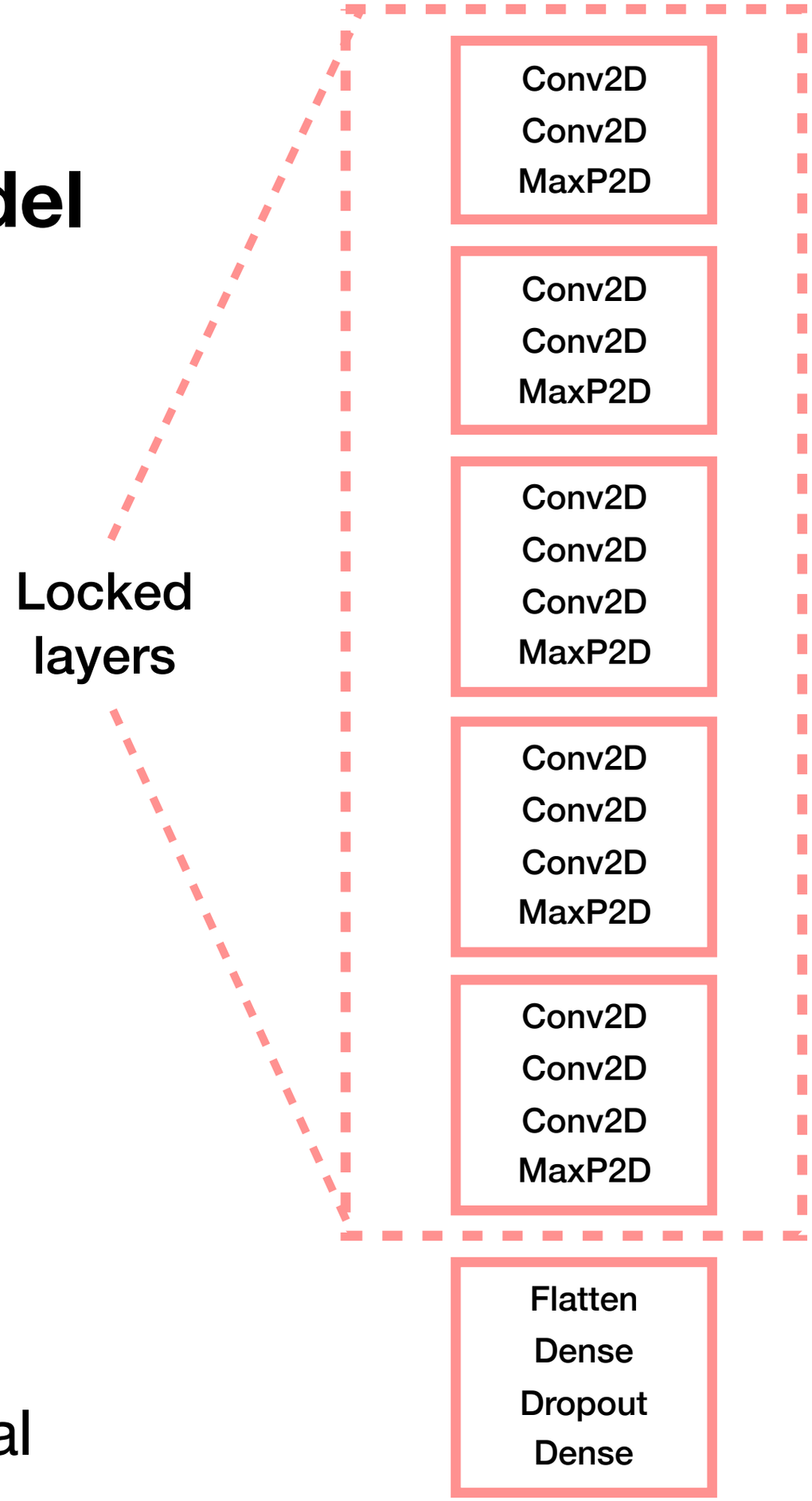
3. Train **model**

Pretrained **filters**

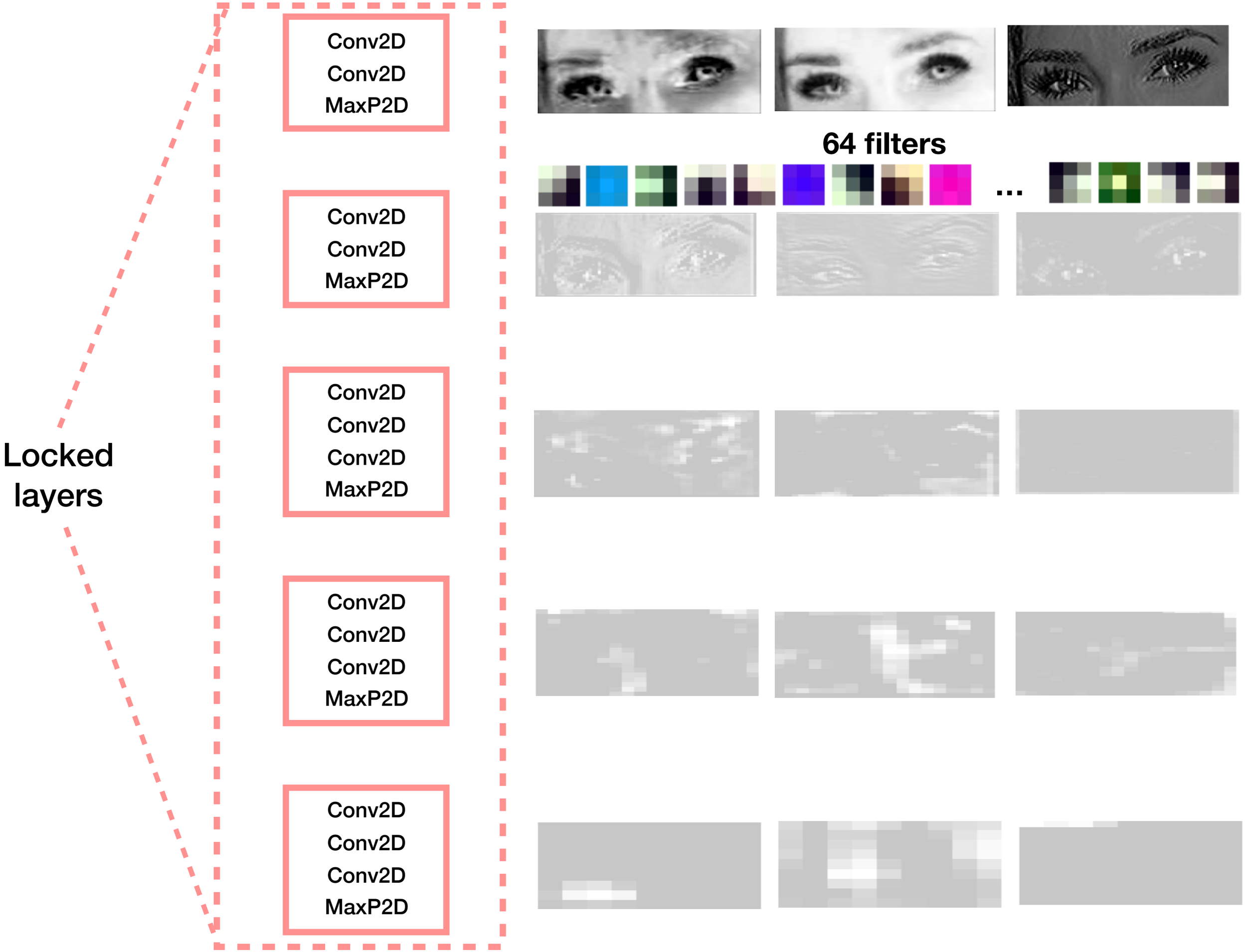


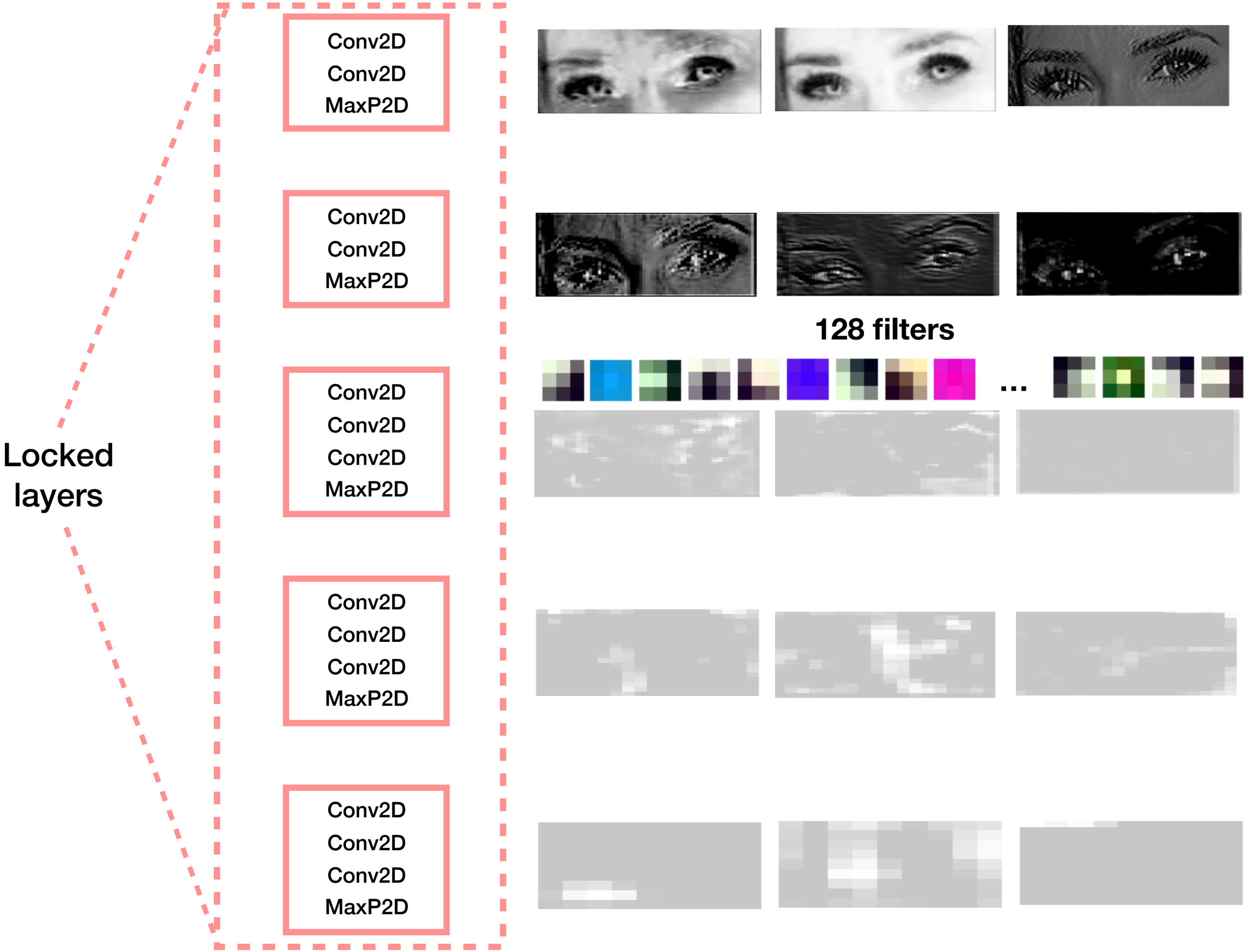
Norah Klintberg Sakal

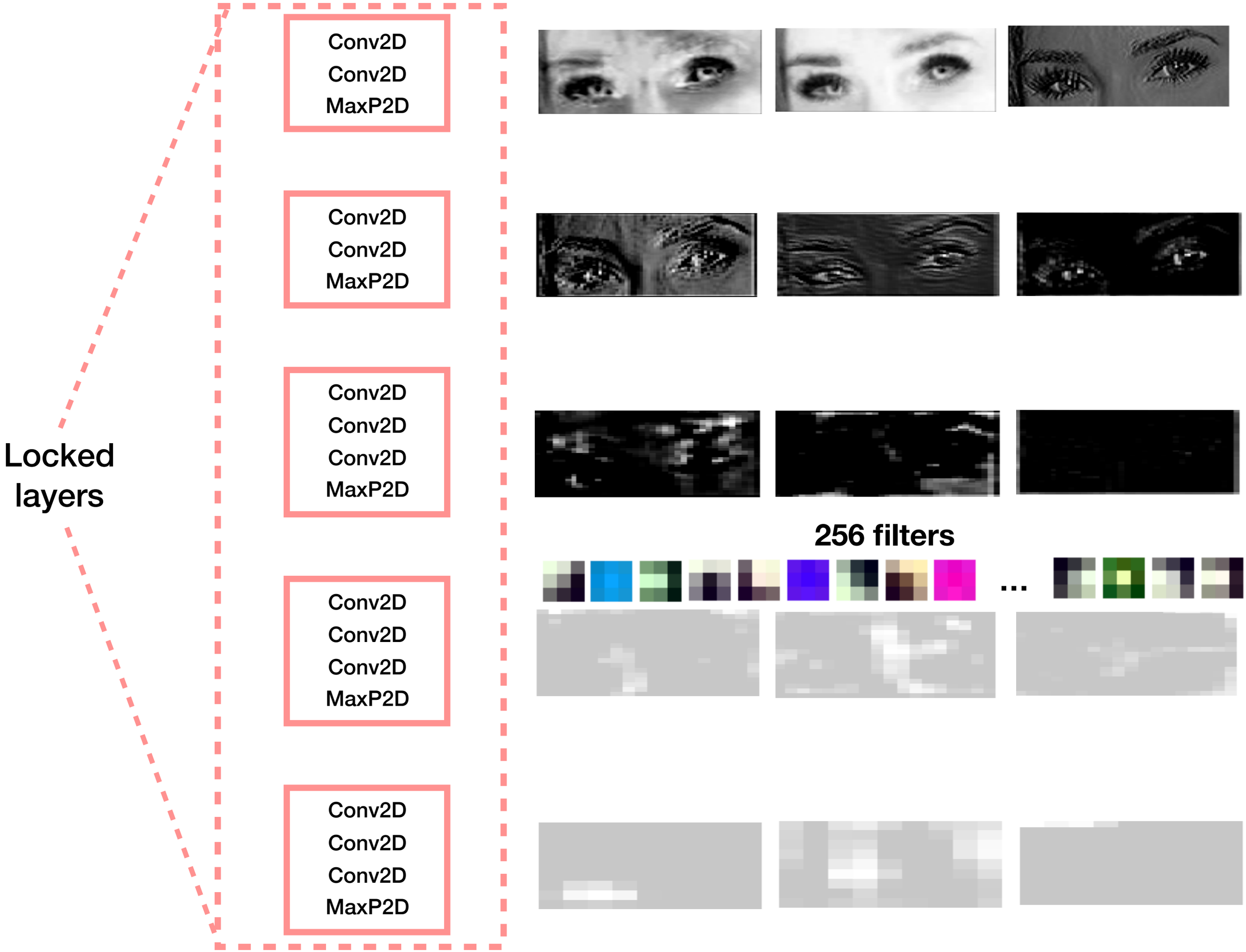
3. Train **model**

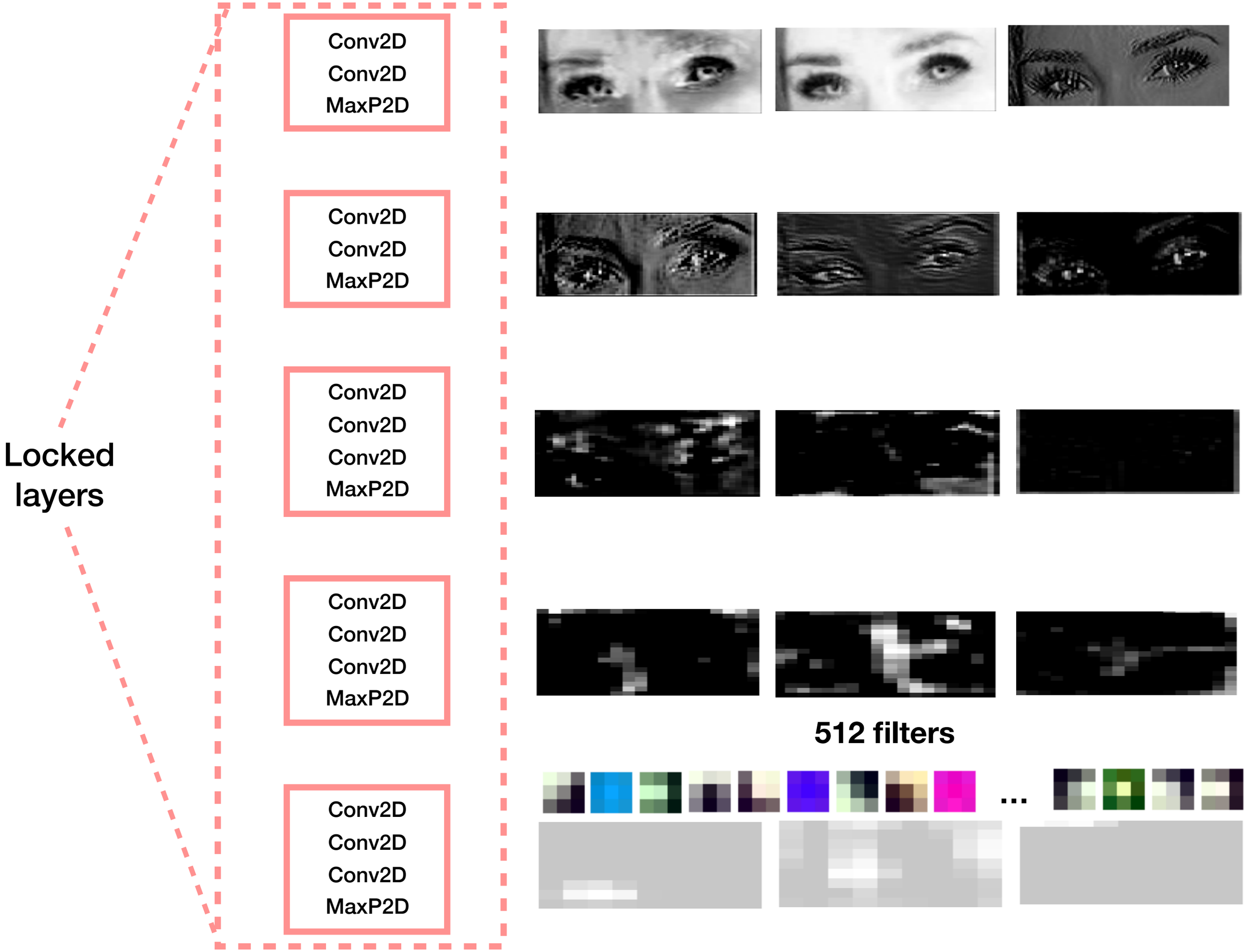


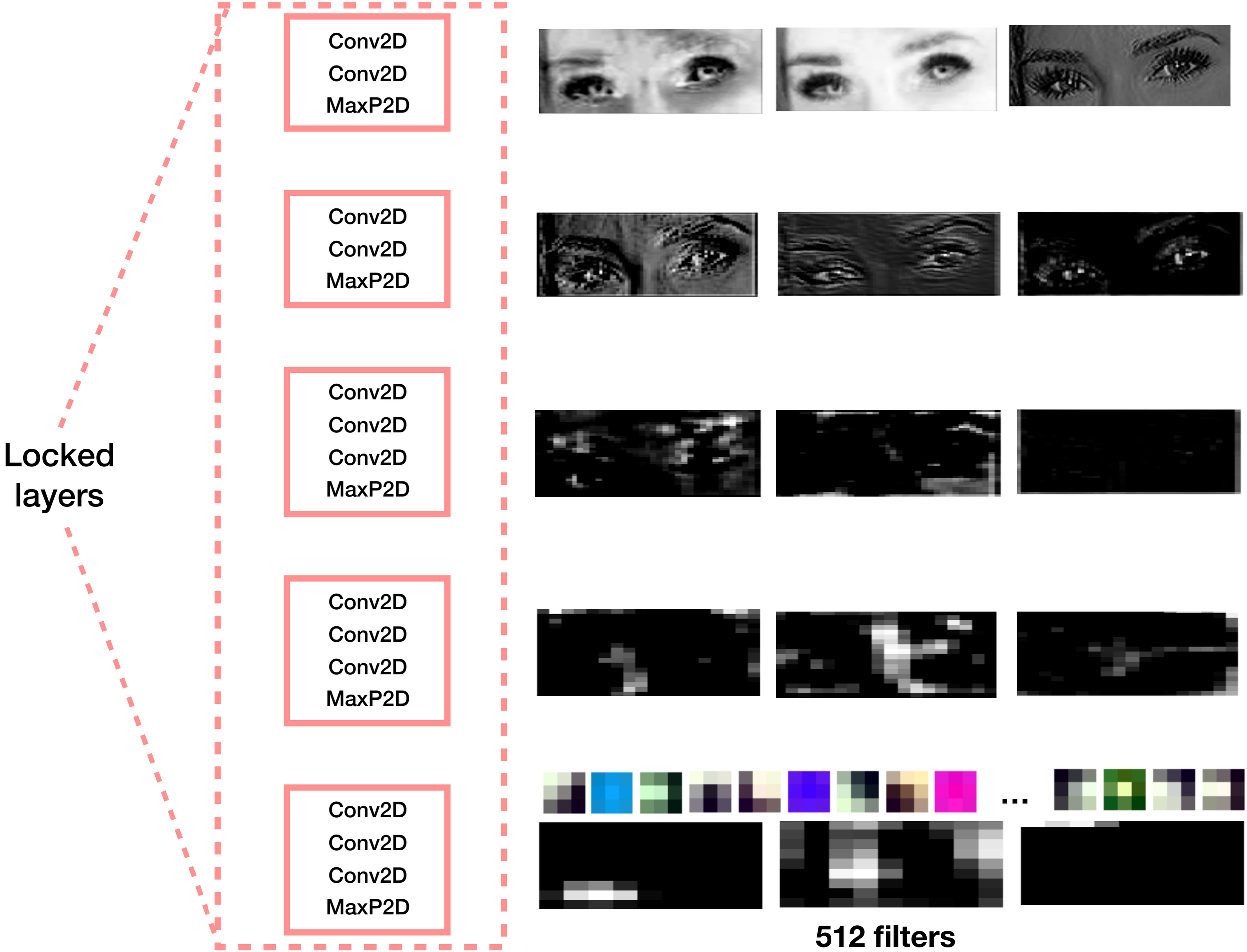
Norah Klintberg Sakal


Locked layers
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
Conv2D
MaxP2D
64 filters
...

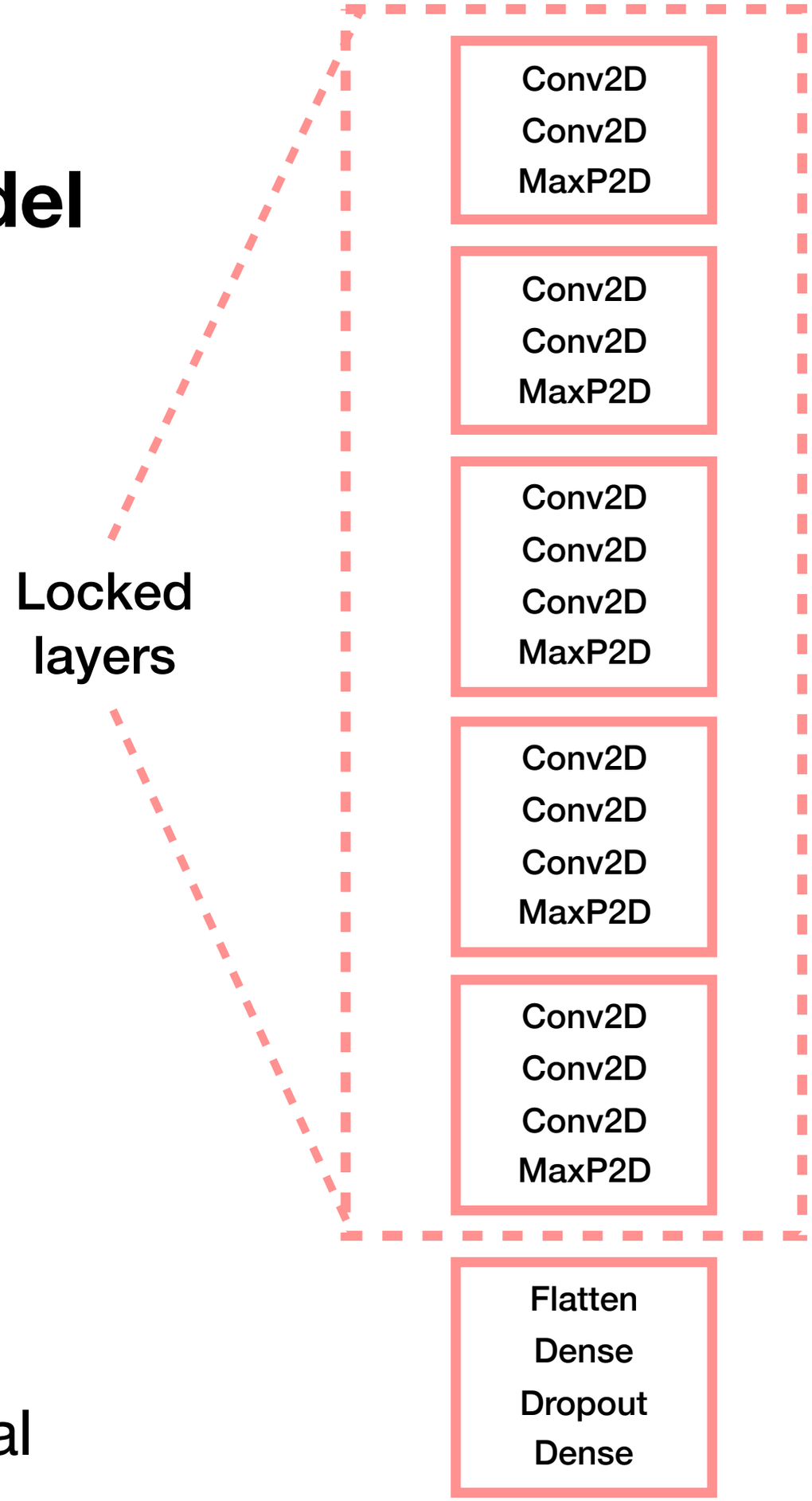

Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
Conv2D
Conv2D
MaxP2D
Locked
layers
128 filters
...


Locked layers
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
Conv2D
MaxP2D
256 filters
...


Locked layers
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
Conv2D
MaxP2D
512 filters
...

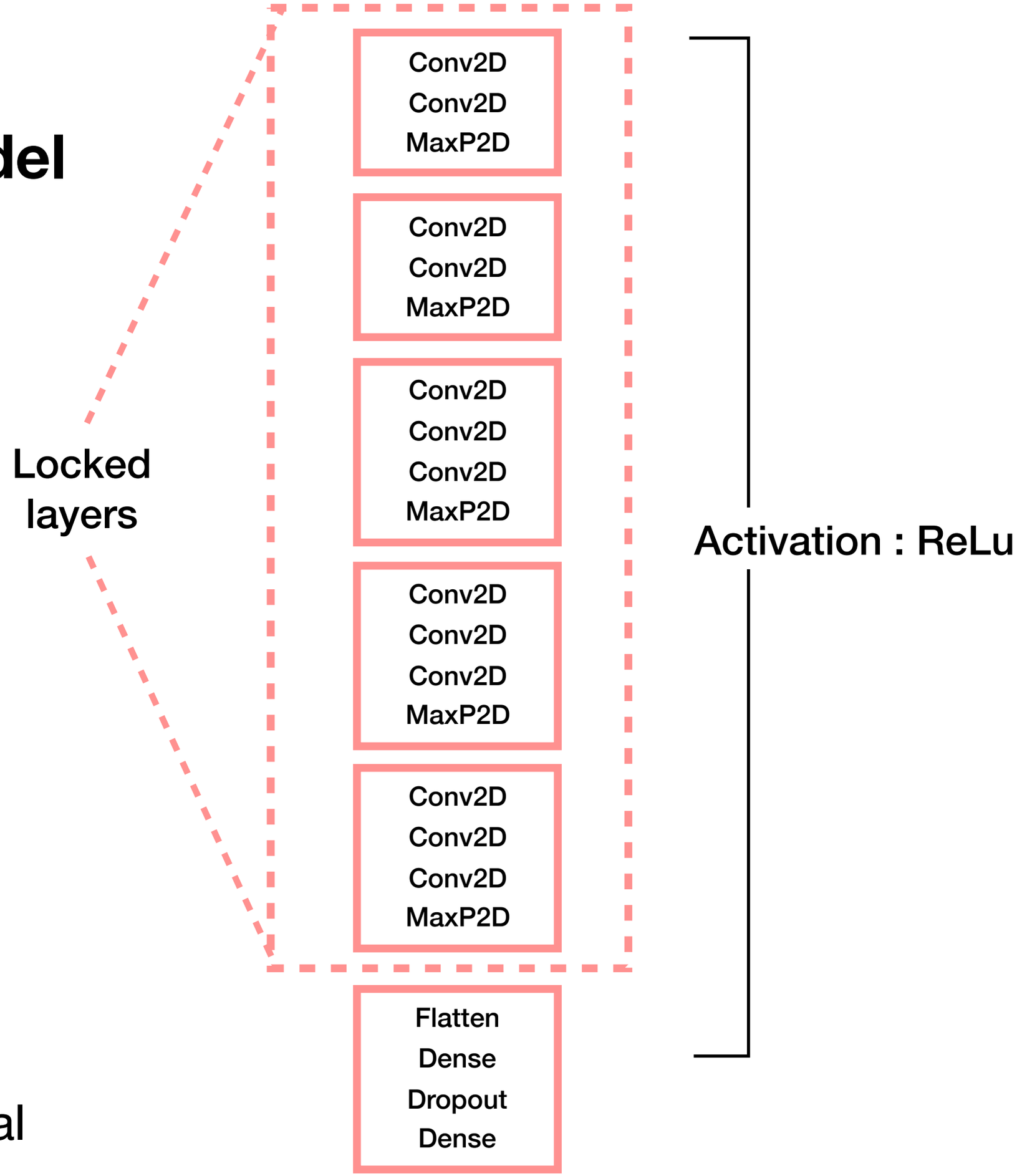

Locked layers
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
Conv2D
MaxP2D
Conv2D
Conv2D
Conv2D
MaxP2D
...
512 filters

3. Train **model**



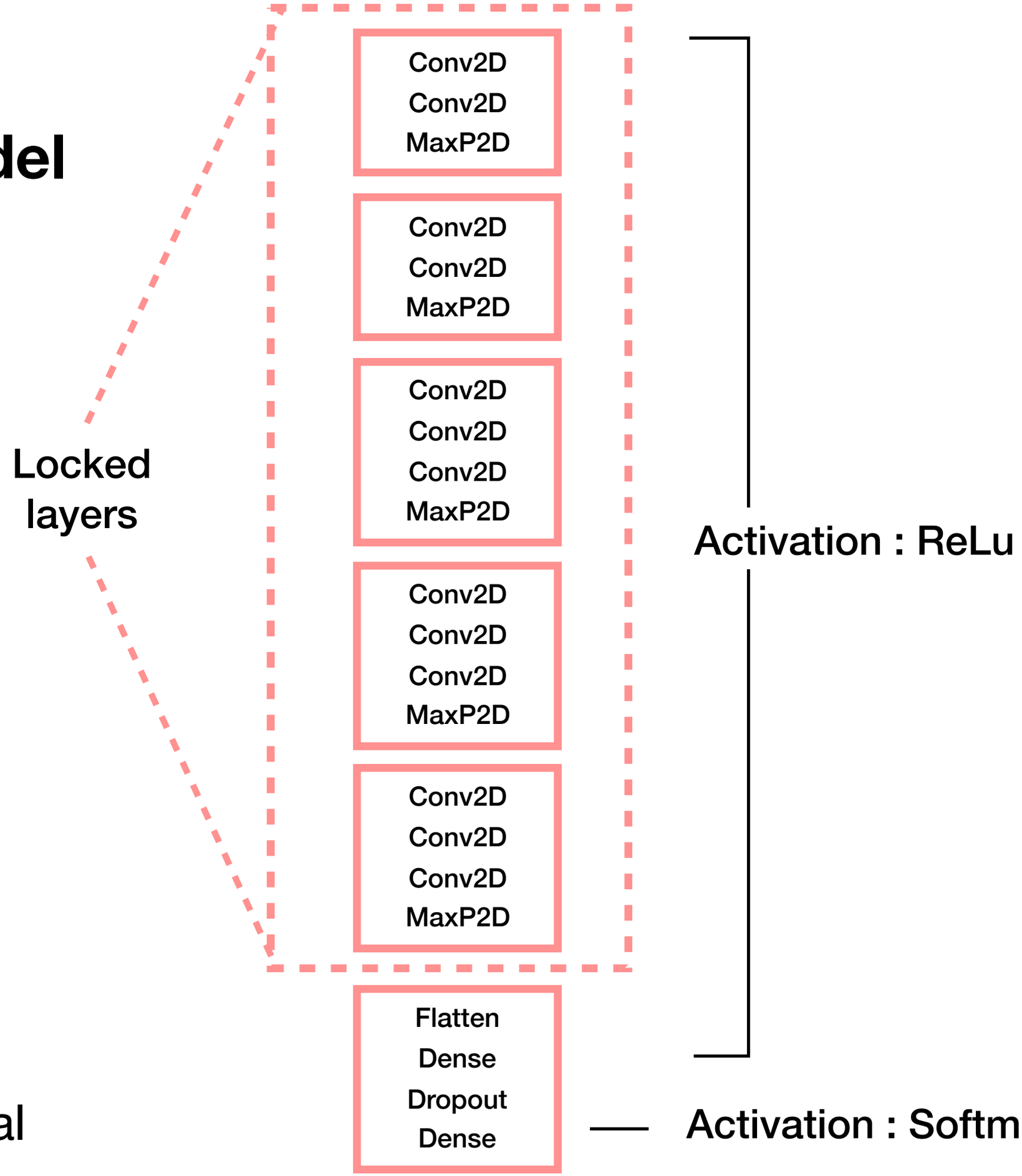
Norah Klintberg Sakal

3. Train **model**



Norah Klintberg Sakal

3. Train **model**



Norah Klintberg Sakal

3. Train **model**

Pre-trained weights: ImageNet
Architecture: VGG16
All conv blocks locked
Activation: ReLu

3. Train **model**

Pre-trained weights: ImageNet
Architecture: VGG16
All conv blocks locked
Activation: ReLu

Dropout
0.5

3. Train **model**

Pre-trained weights: ImageNet
Architecture: VGG16
All conv blocks locked
Activation: ReLu

Dropout
0.5

Last fully connected
Activation: Softmax

3. Train **model**

Pre-trained weights: ImageNet
Architecture: VGG16
All conv blocks locked
Activation: ReLu

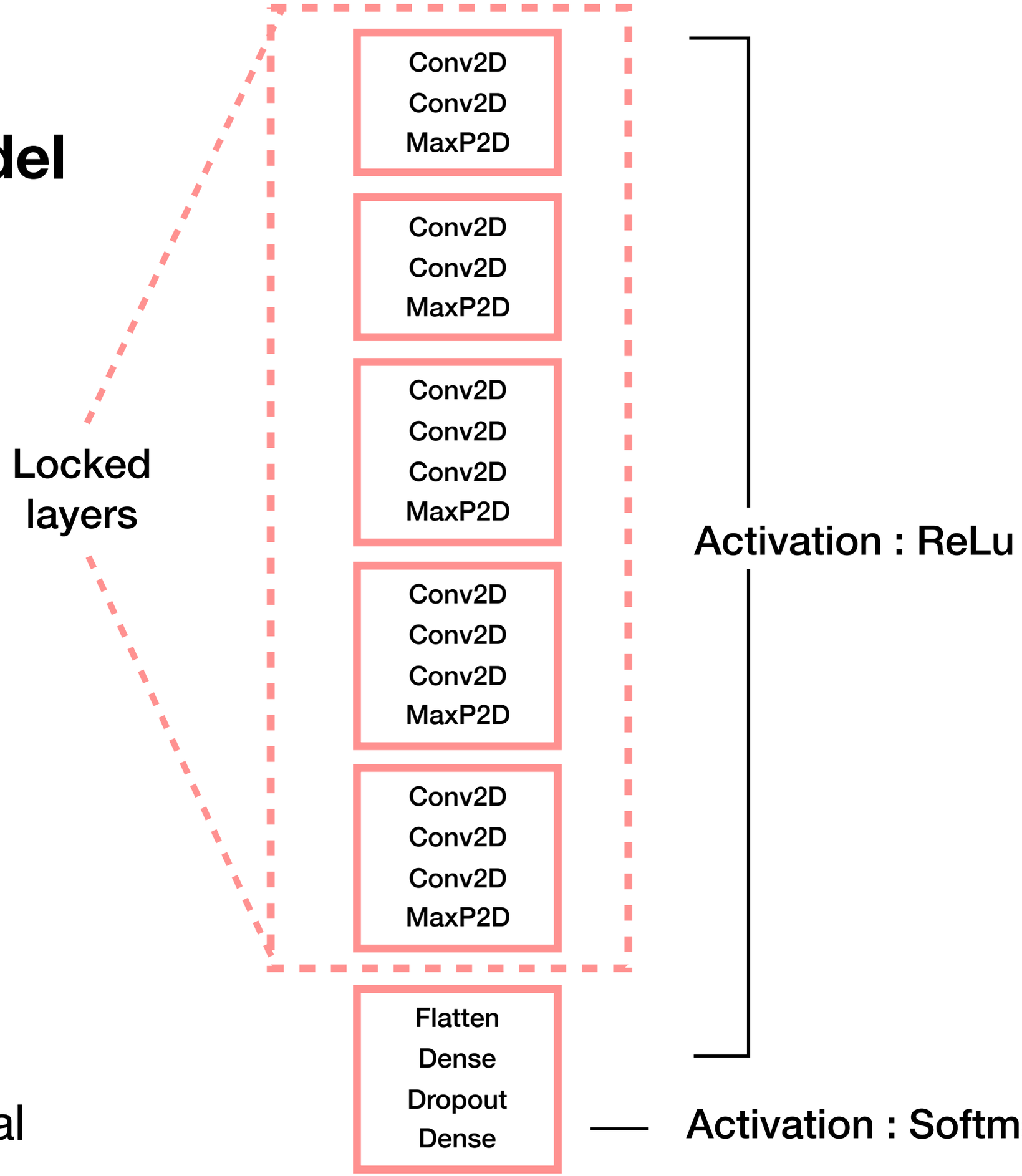
Dropout
0.5

Last fully connected
Activation: Softmax

Epochs: 50
Batch size: 16

Norah Klintberg Sakal

3. Train **model**



Norah Klintberg Sakal

Training module

In [ ]:
```
base_model= VGG16(weights='imagenet', include_top=False,
                  input_shape=(150,150,3))
```

In [ ]:
```
top_model = Flatten()(base_model_functional.output)
top_model = Dense(256, activation='relu')(top_model)
top_model = Dropout(0.5)(top_model)
top_model = Dense(classes, activation='softmax')(top_model)
```

In [ ]:
```
model = Model(inputs=base_model.input, outputs=top_model)
```

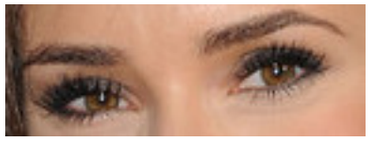
In [ ]:
```
for layer in model.layers[:19]:
    layer.trainable = False
```

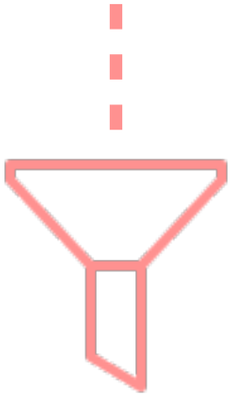
In [ ]:
```
model.compile(loss=loss,
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [ ]:
```
model.fit_generator()
```

In [ ]:
```
model.save_weights('my_model.h5')
```

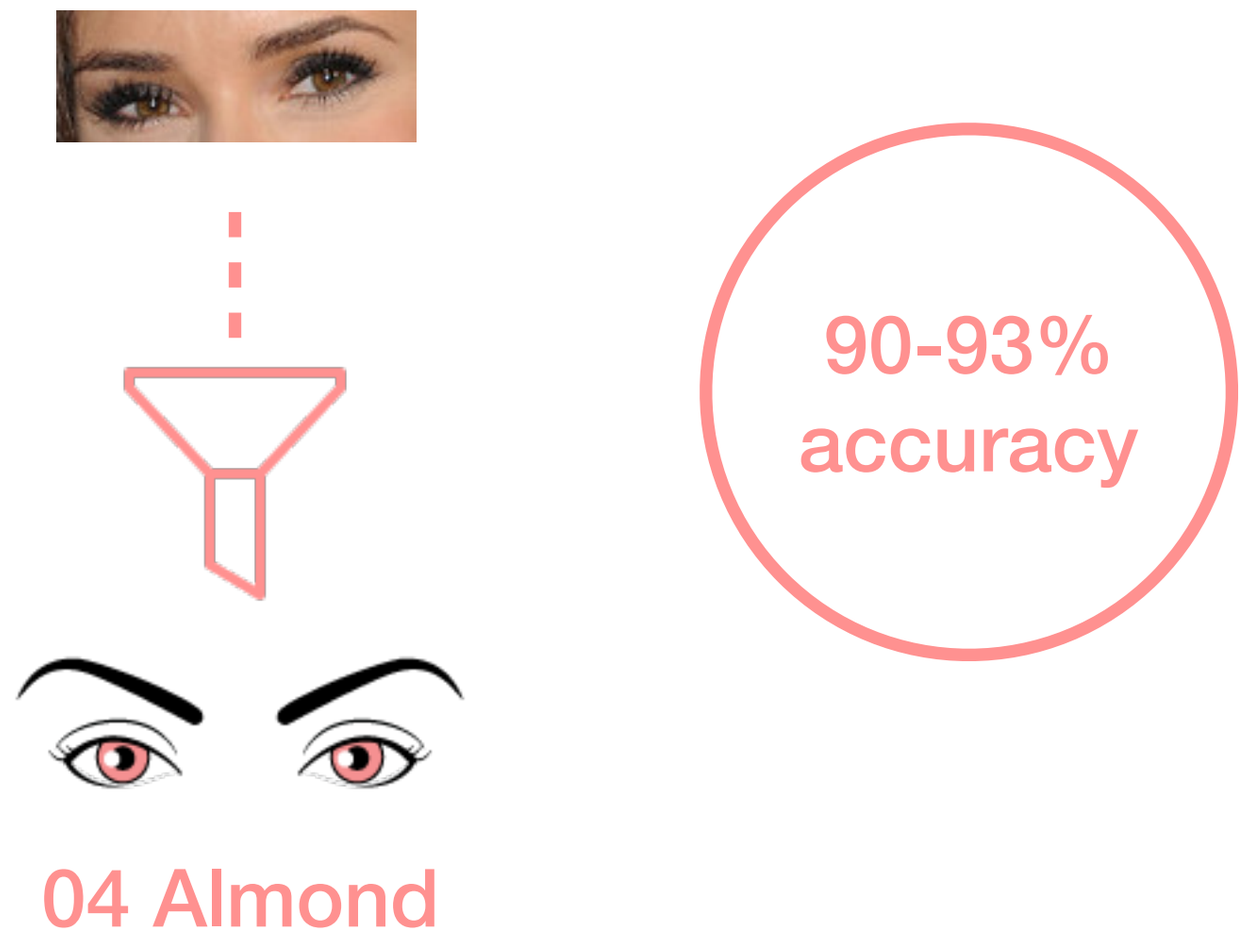
3. Train **model**



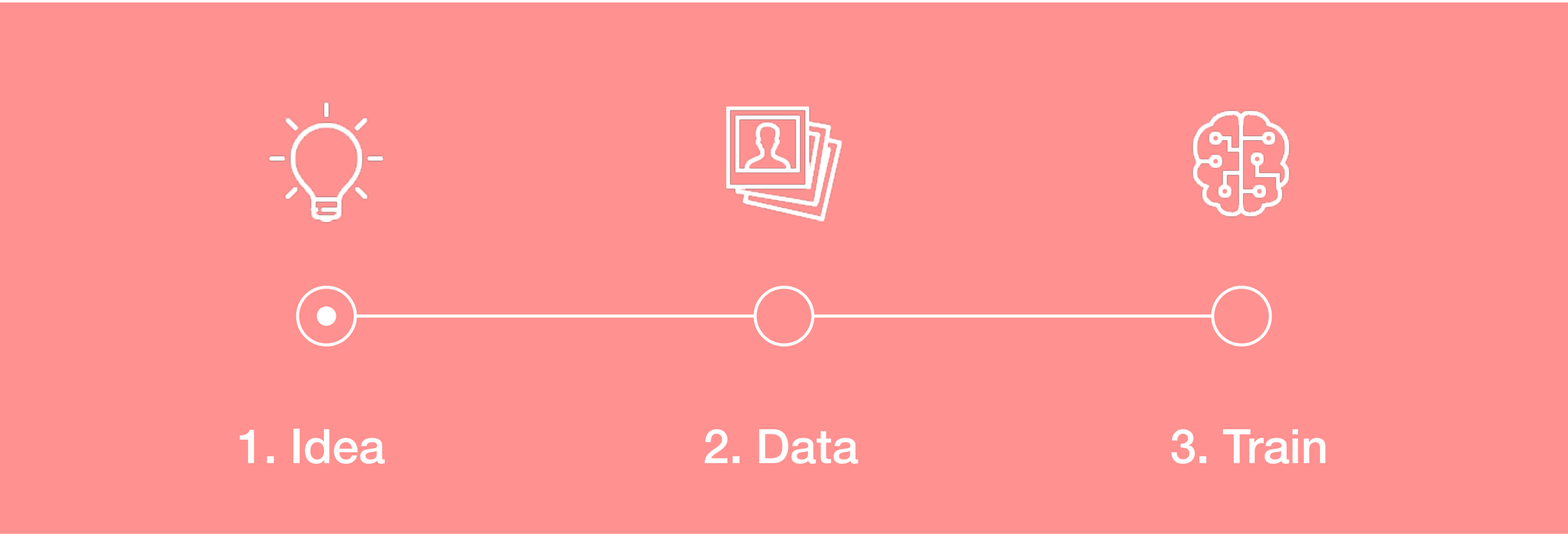


Accuracy 0-100% ?

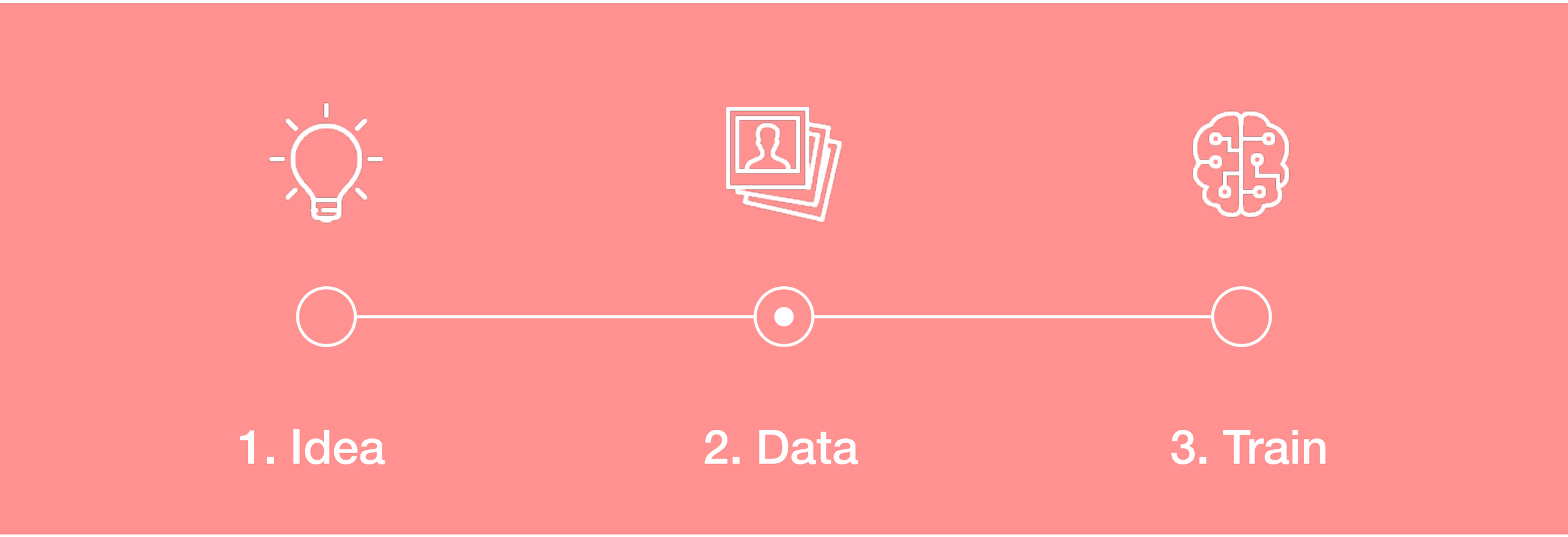
3. Train **model**



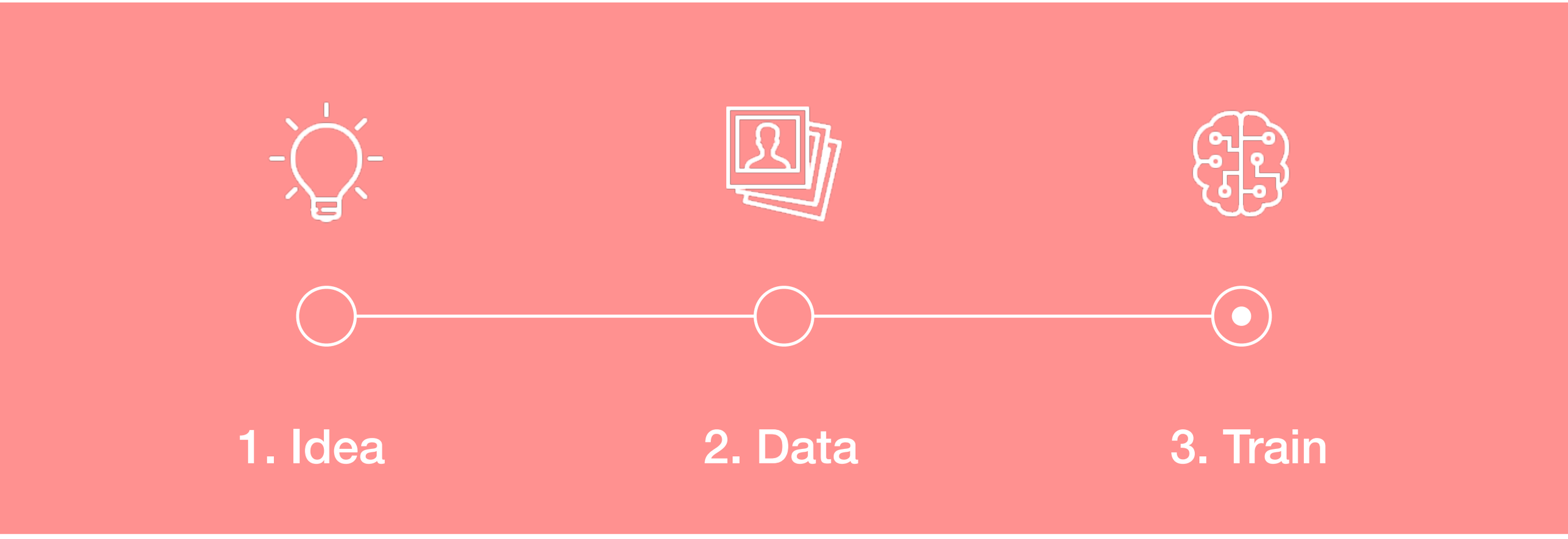
3 steps



3 steps



3 steps

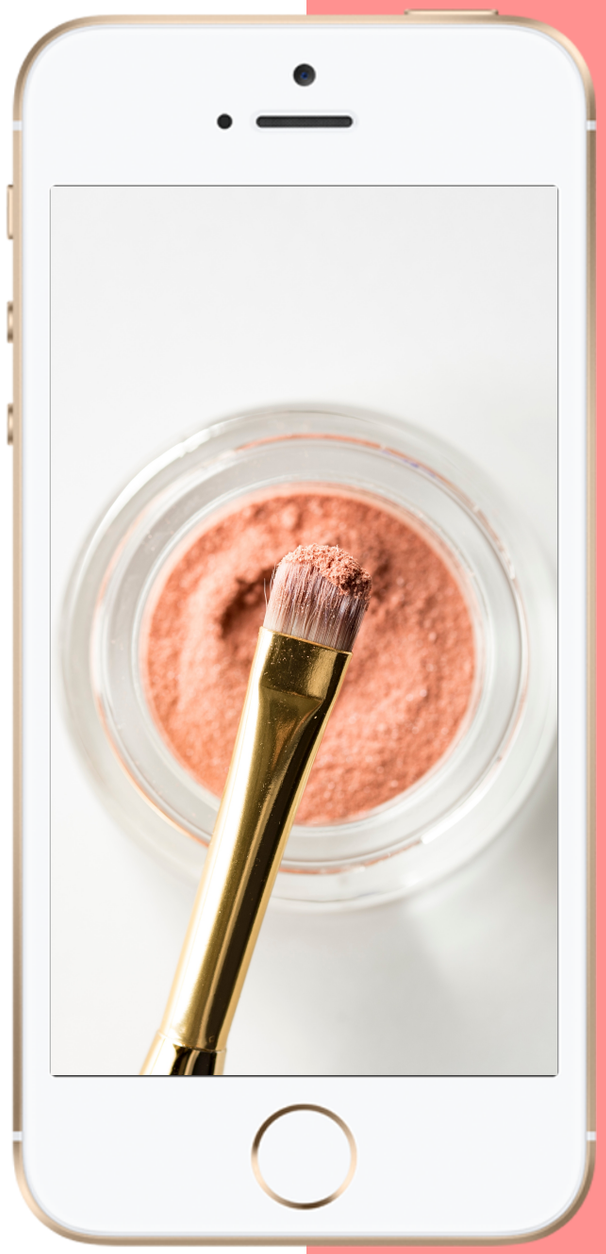


App

01. Frontend

02. Backend

03. Web server
- Flask



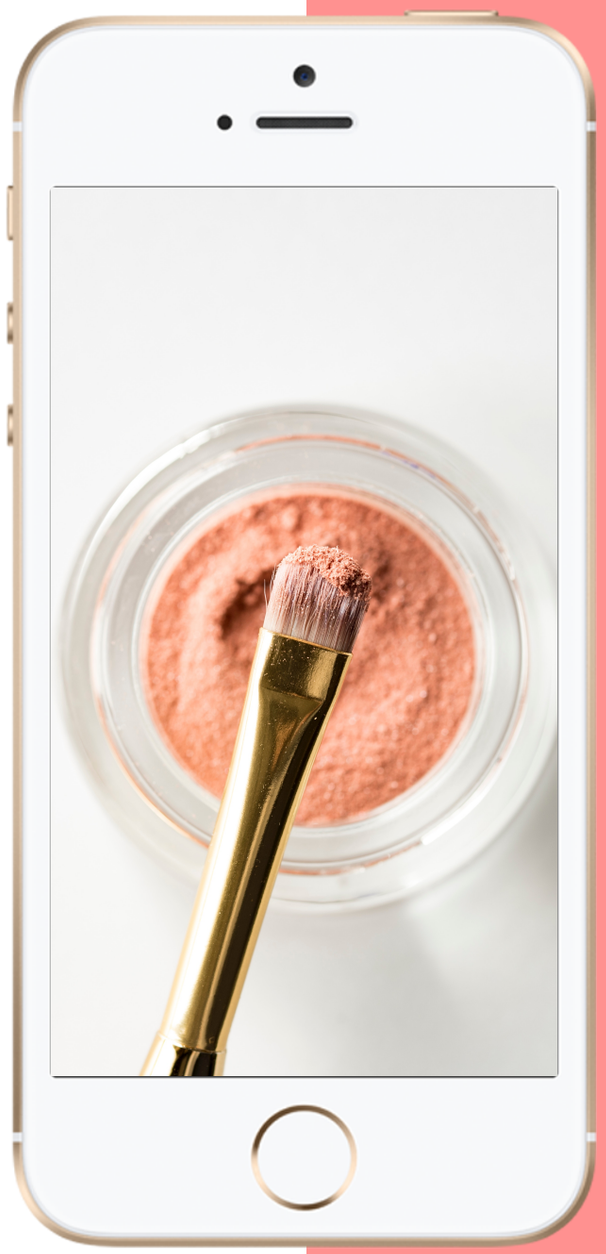
Norah Klintberg Sakal

App

01. Frontend

02. Backend

03. Web server
- Flask

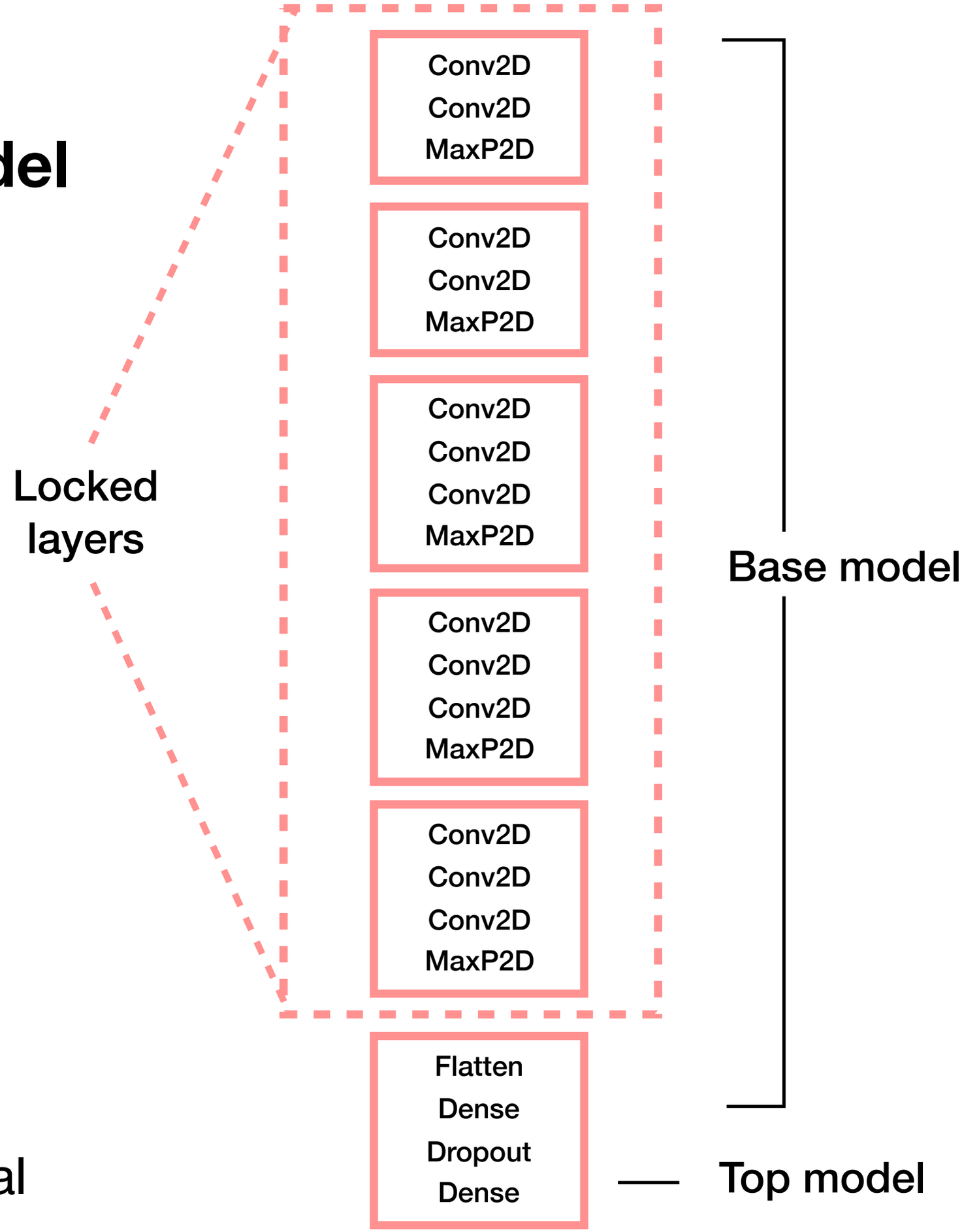


Norah Klintberg Sakal

Load model

```
base_model= VGG16(weights='imagenet', include_top=False,
                  input_shape=(150,150,3))

top_model = Flatten()(base_model_functional.output)
top_model = Dense(256, activation='relu')(top_model)
top_model = Dropout(0.5)(top_model)
top_model = Dense(classes, activation='softmax')(top_model)

model = Model(inputs=base_model.input, outputs=top_model)

model.load_weights(weights_path)

model.compile(loss=loss,
              optimizer='rmsprop',
              metrics=['accuracy'])

```

3. Train **model**



Norah Klintberg Sakal

Load model

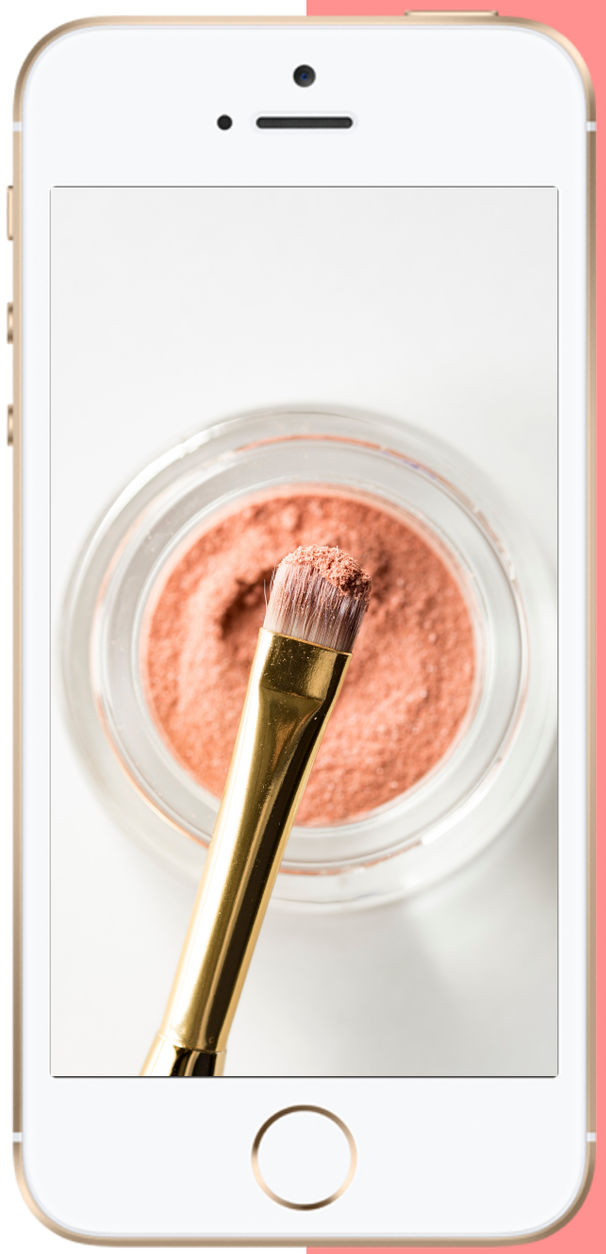
```
base_model= VGG16(weights='imagenet', include_top=False,
                  input_shape=(150,150,3))

top_model = Flatten()(base_model_functional.output)
top_model = Dense(256, activation='relu')(top_model)
top_model = Dropout(0.5)(top_model)
top_model = Dense(classes, activation='softmax')(top_model)

model = Model(inputs=base_model.input, outputs=top_model)

model.load_weights(weights_path)

model.compile(loss=loss,
              optimizer='rmsprop',
              metrics=['accuracy'])

```

App

01. Frontend

02. Backend

03. Web server
- Flask



Norah Klintberg Sakal

Web server

```python
@app.route('/predictImage', methods=['POST'])
def predictImage():
    img = image.load_img(img_path, target_size=(150, 150))
    x = image.img_to_array(img)

    preds = model.predict_classes(x)
    classes = {2:'ME', 3:'RE', 0:'AE', 1:'HE'}

    return classes[preds[0]]


if __name__ == "__main__":
    app.run(debug=True,port=5555)
```

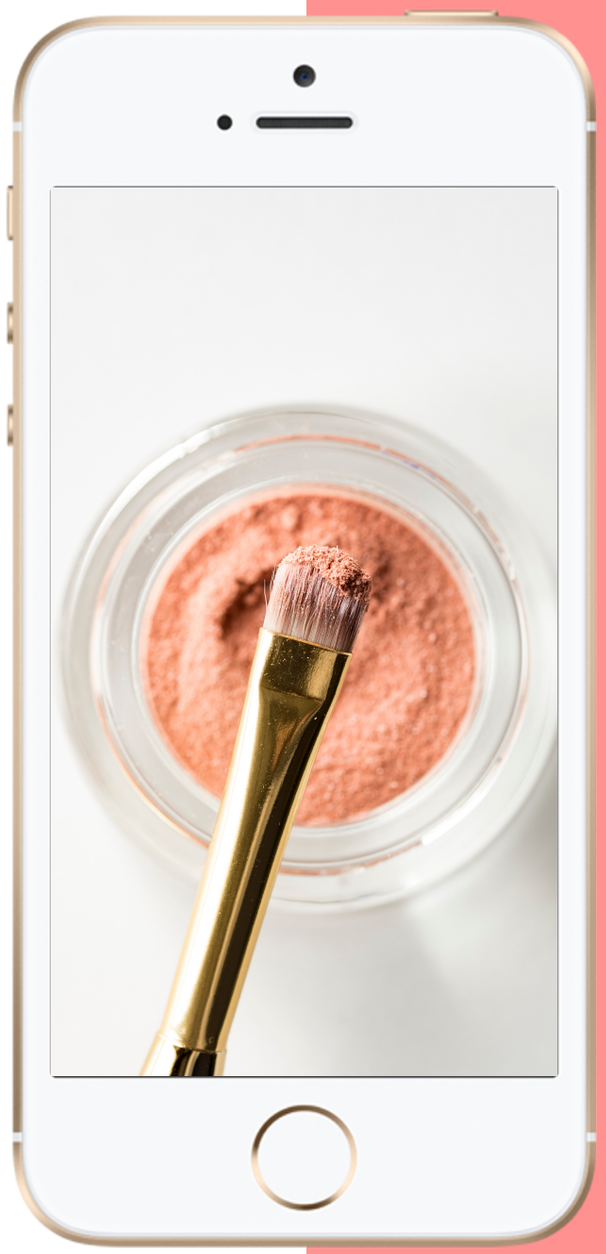
- Trained model

App

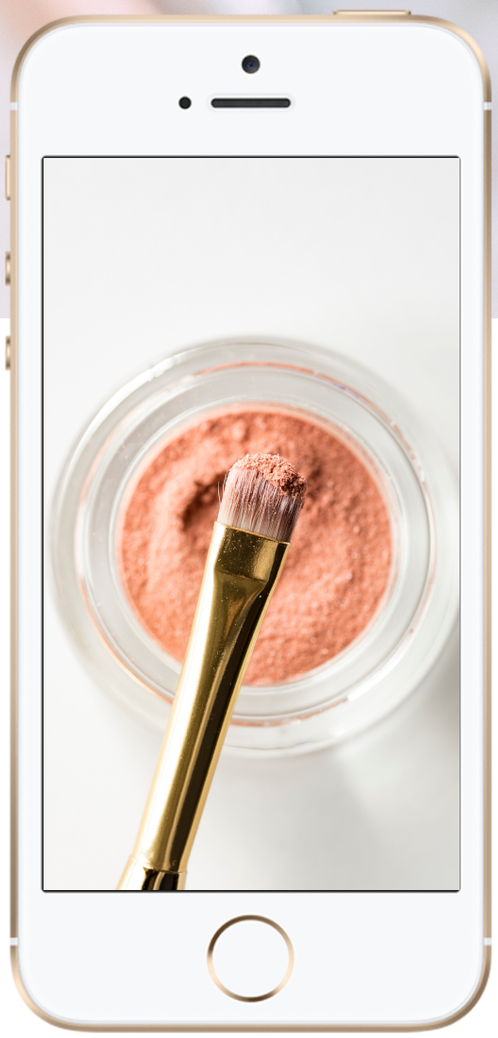
01. Frontend

02. Backend

03. Web server
- Flask



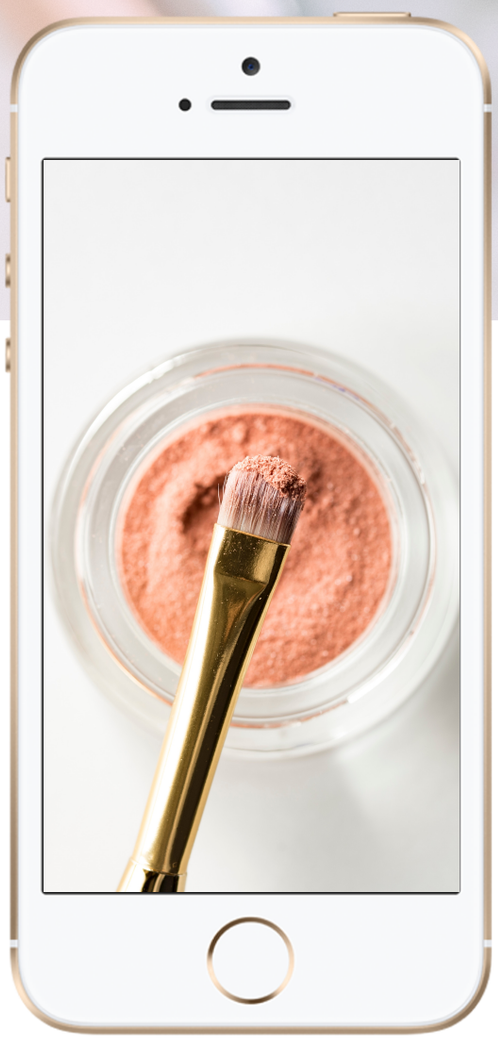
Norah Klintberg Sakal



1 Frontend

2 Backend

1. Take photo
2. Upload photo
3. Predicts image
4. Your eye shape



1 Front**end**

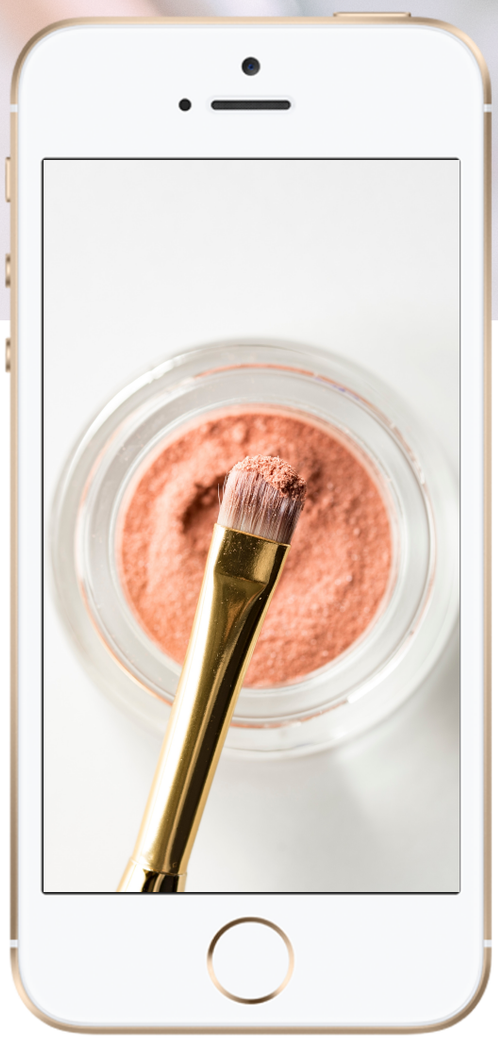
2 Back**end**

1. Take photo

2. Upload photo

3. Predicts image

4. Your eye shape



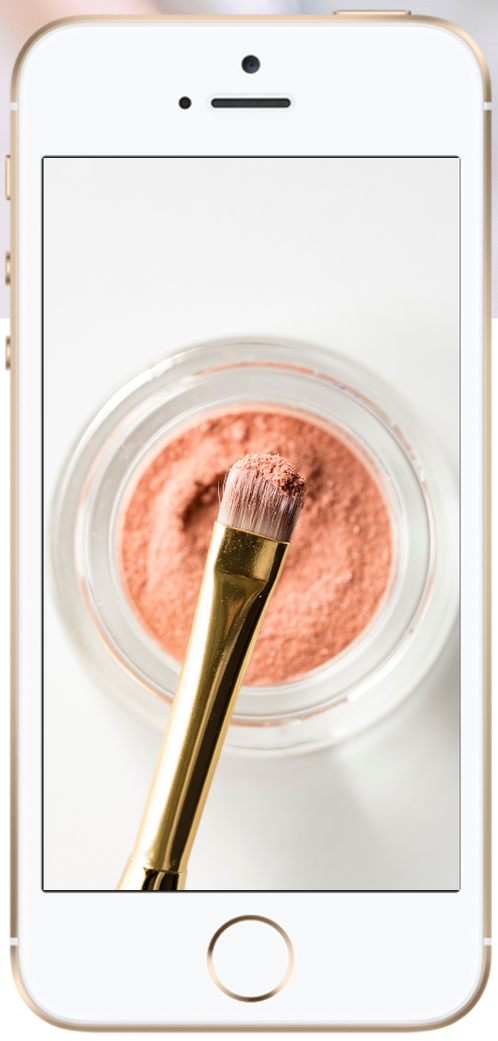
1 Front**end**

2 Back**end**

1. Take photo

2. Upload photo → 3. Predicts image

4. Your eye shape



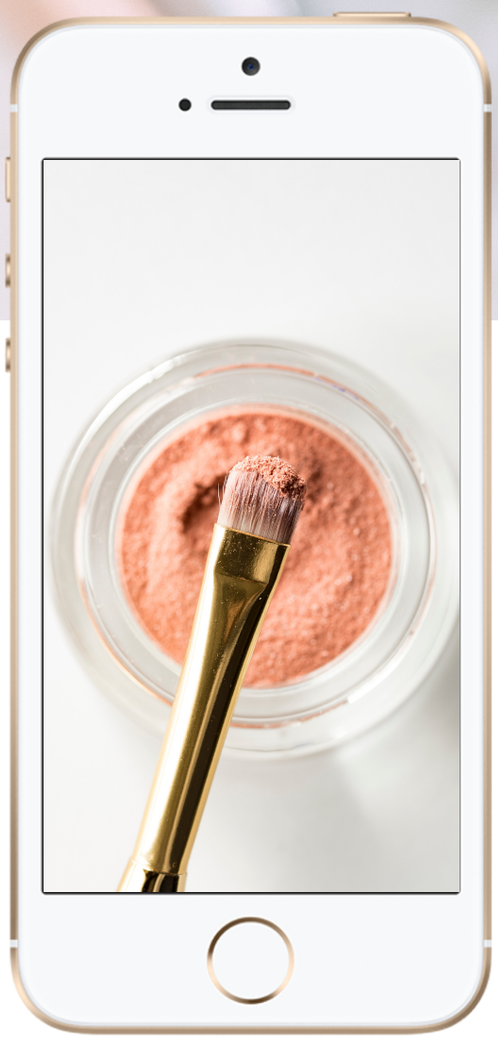
1 Frontend

2 Backend

1. Take photo

2. Upload photo → 3. Predicts image

4. Your eye shape



1 Front**end**

2 Back**end**

1. Take photo

2. Upload photo → 3. Predicts image

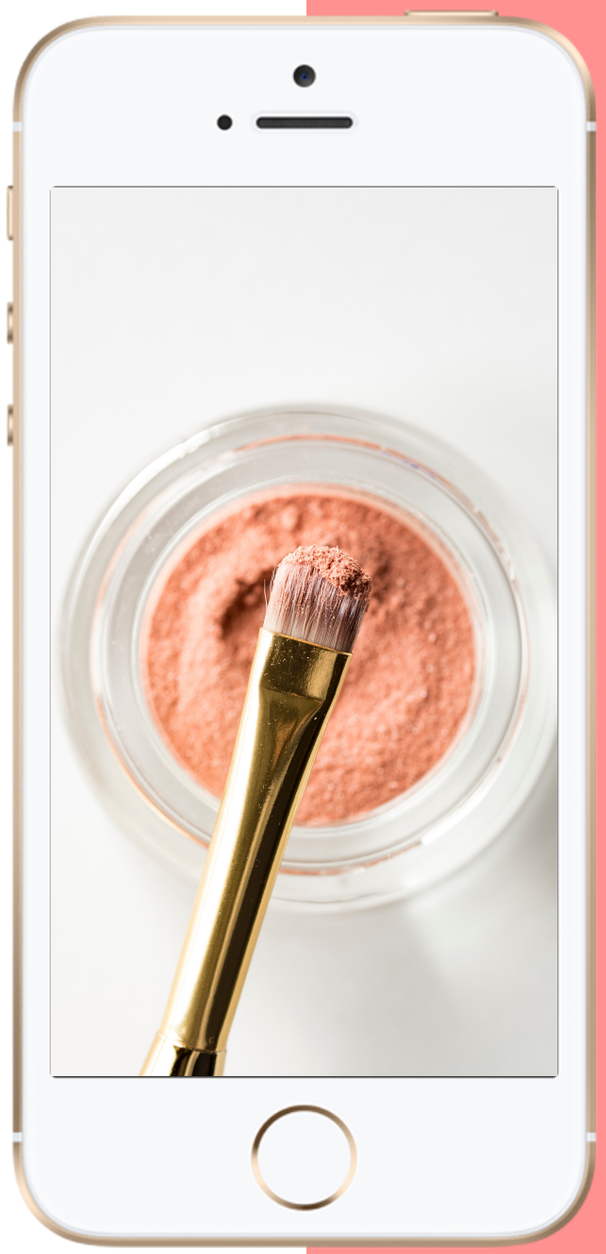
4. Your eye shape

App

01. Frontend

02. Backend

03. Web server
- Flask



Norah Klintberg Sakal

Repository norahsakal

norahsakal / pycascades-2019-shades

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

Train your own deep learning algorithm with Jupyter notebook and then predict on your newly trained model with a React frontend and a Python Flask backend

Edit

Manage topics

2 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

norahsakal Working app — Latest commit b12a945 3 minutes ago

frontend	Working app	3 minutes ago
README.md	Working app	3 minutes ago
app.py	Working app	3 minutes ago
requirements.txt	Working app	3 minutes ago
train_your_model.ipynb	Working app	3 minutes ago

README.md

Guide to your own artificial intelligence app in 3 steps

1) Clone repo and install requirements from requirements.txt

Repository **walkthrough**

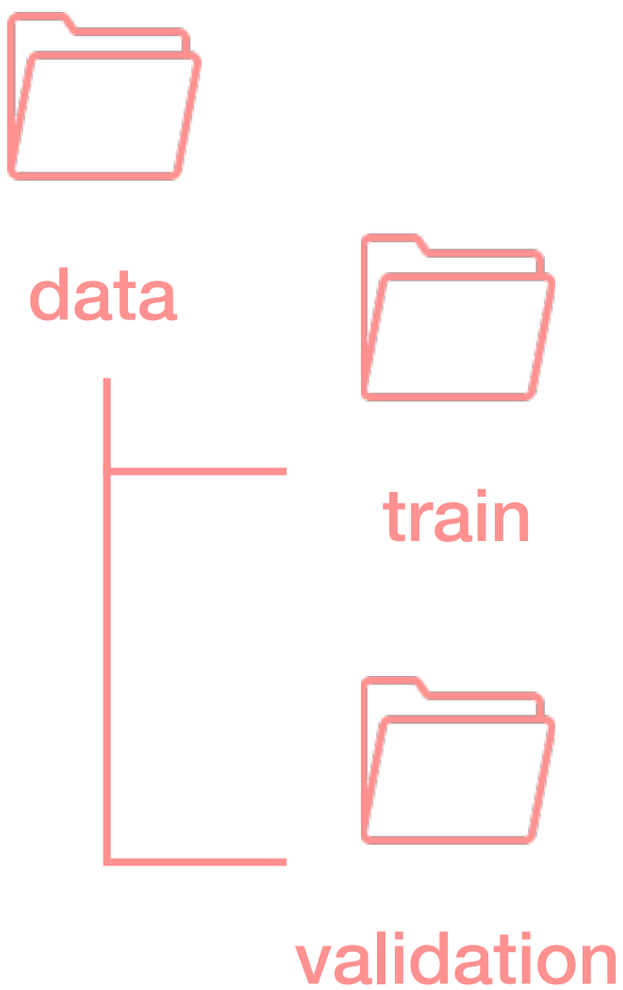
01. Train a model - Jupyter notebook

02. Save your model - Jupyter notebook

03. Test your model - React.js frontend

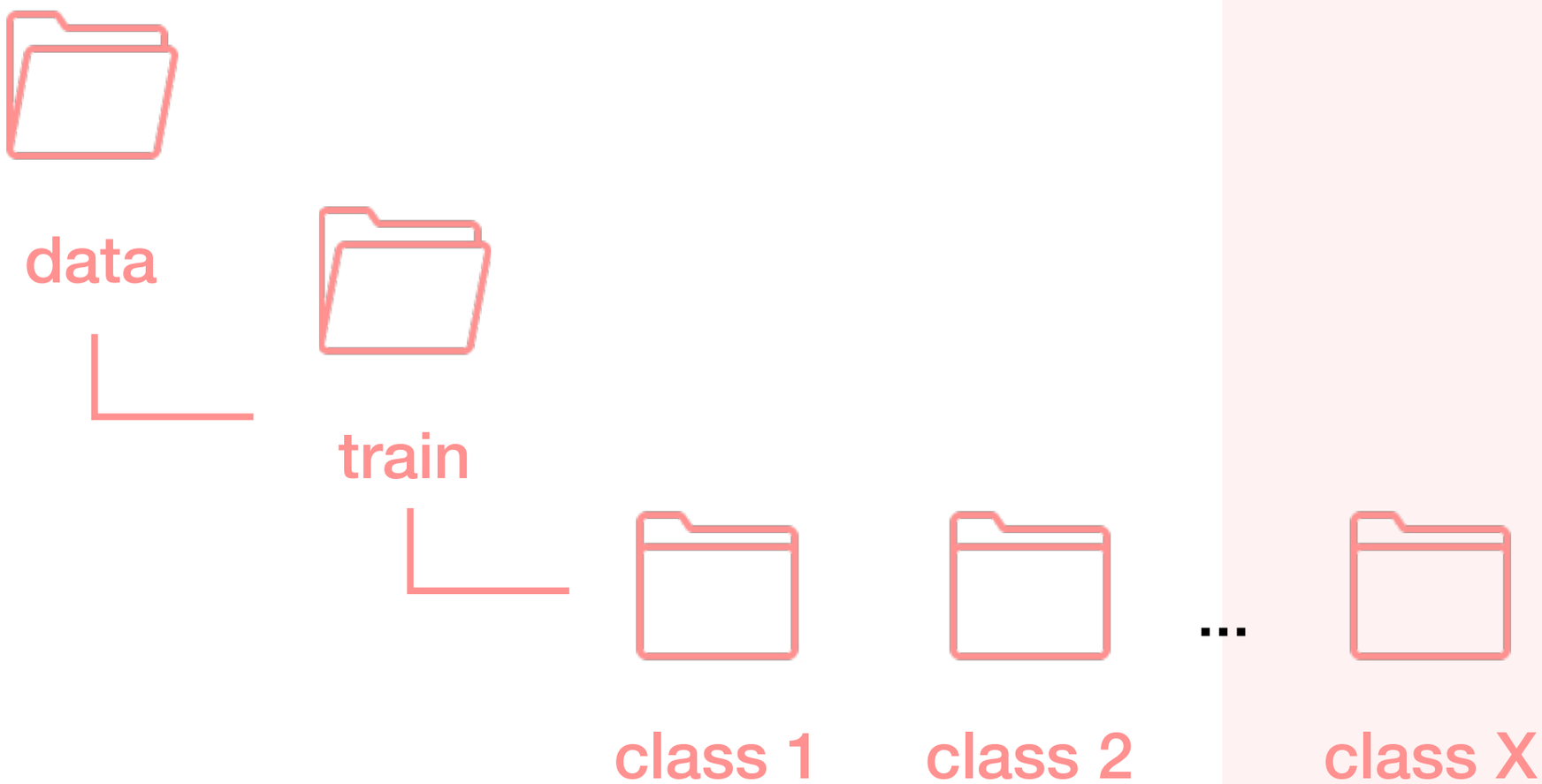
Repository **walkthrough**

01. Train a model - Jupyter notebook



Repository **walkthrough**

01. Train a model - Jupyter notebook

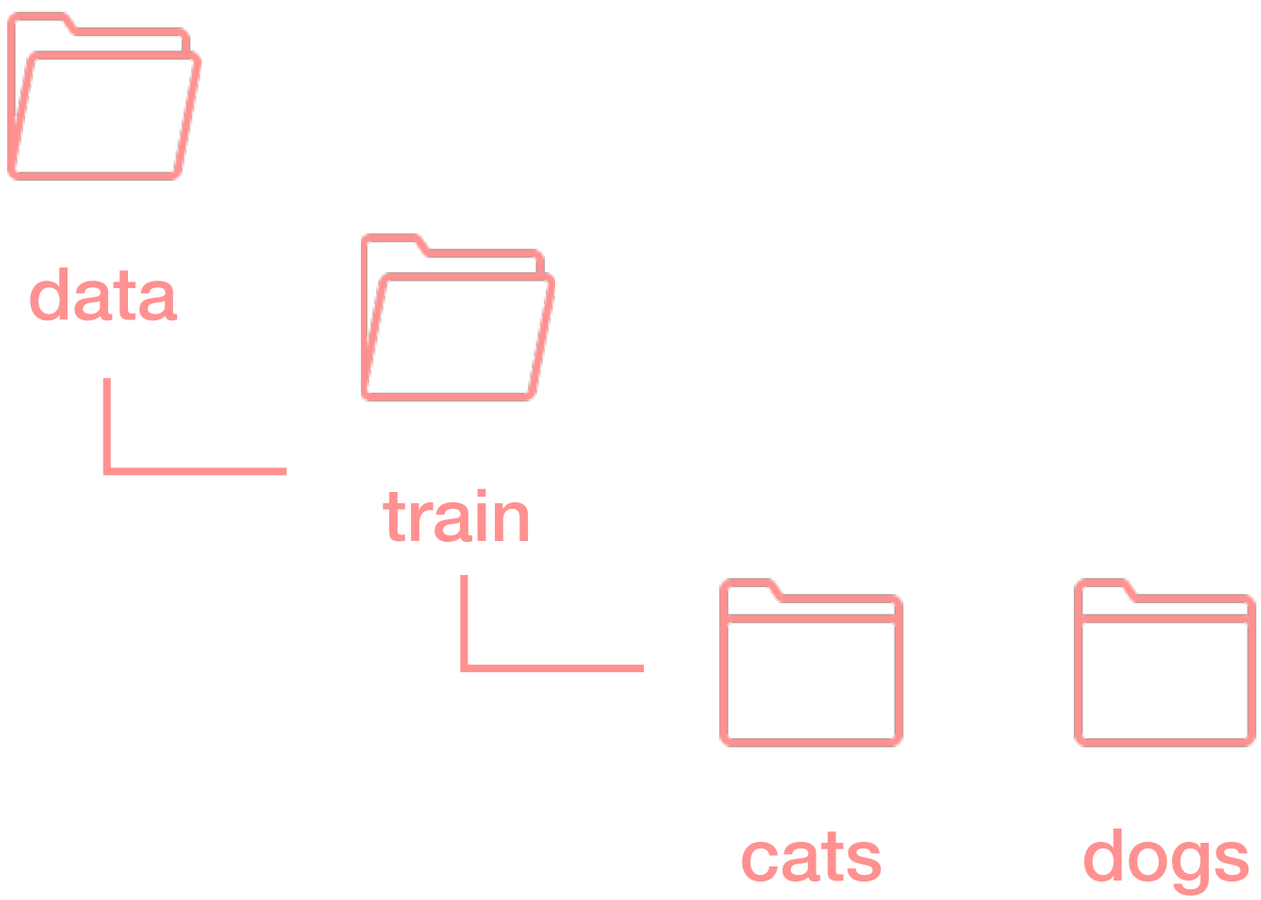


Repository **walkthrough**

01. Train a model - Jupyter notebook

Repository **walkthrough**

01. Train a model - Jupyter notebook

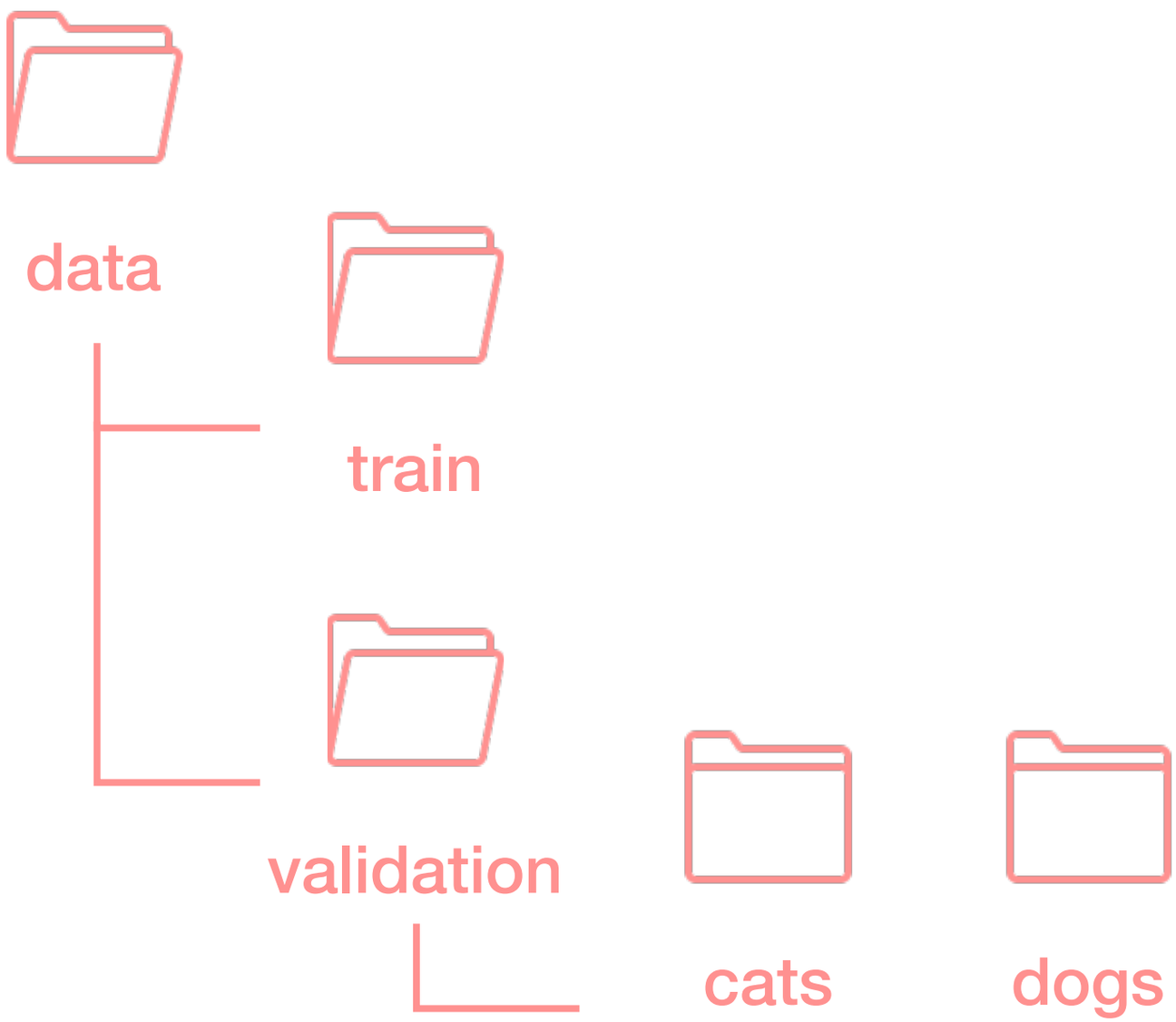


Repository **walkthrough**

01. Train a model - Jupyter notebook

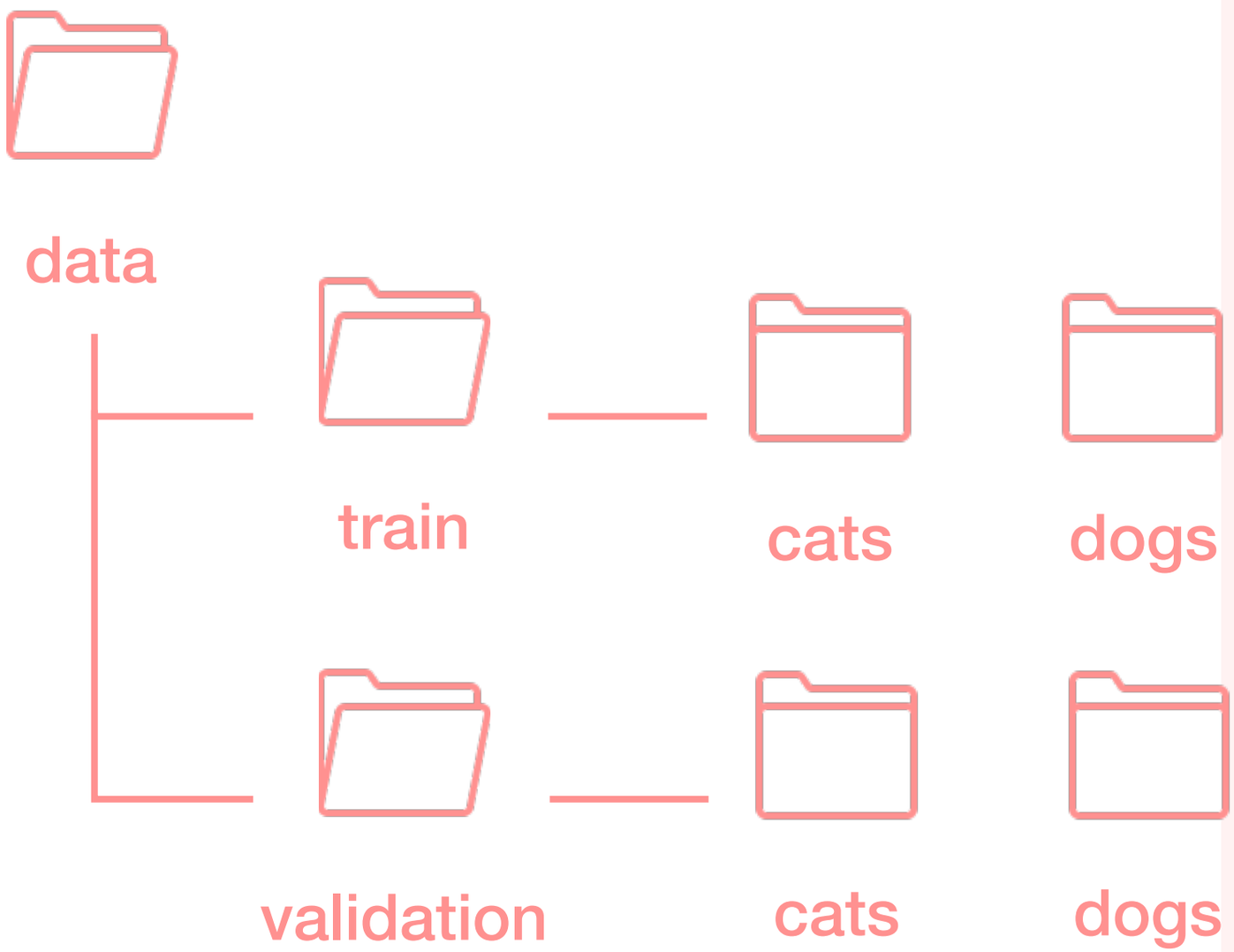
Repository **walkthrough**

01. Train a model - Jupyter notebook



Repository **walkthrough**

01. Train a model - Jupyter notebook



Repository walkthrough

01. Train a model - Jupyter notebook

02. Save your model - Jupyter notebook

```
model.save_weights('your_model.h5')
```

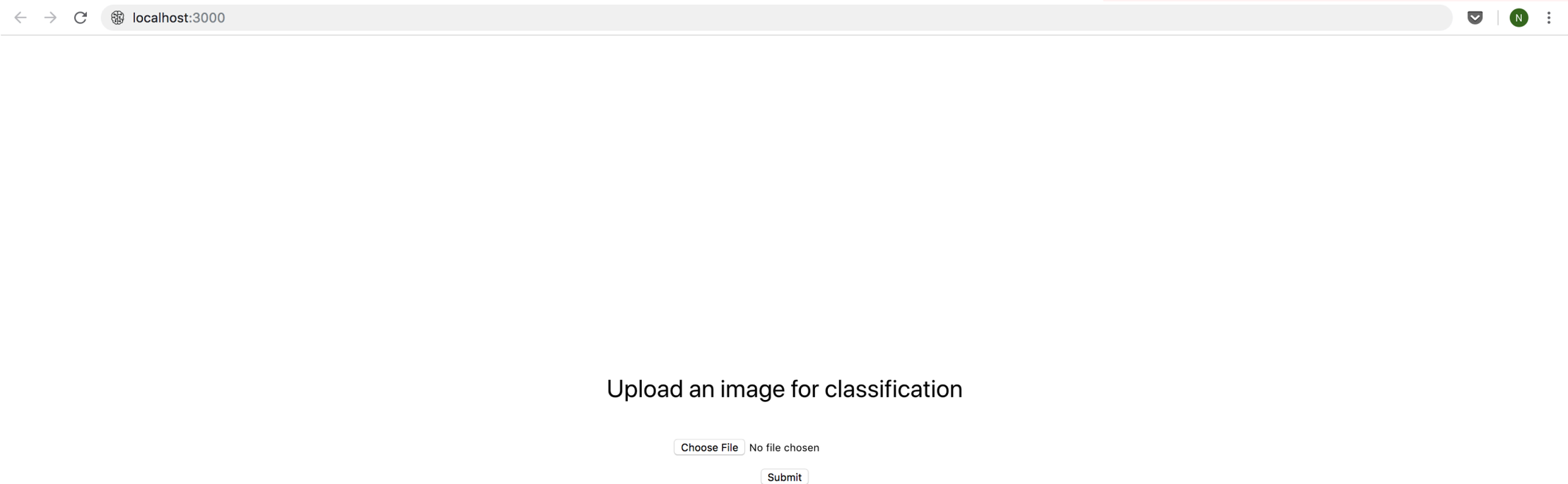
Repository walkthrough

01. Train a model - Jupyter notebook

02. Save your model - Jupyter notebook

03. Test your model - React.js frontend

Repository walkthrough



Repository norahsakal

norahsakal / pycascades-2019-shades

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

Train your own deep learning algorithm with Jupyter notebook and then predict on your newly trained model with a React frontend and a Python Flask backend

Edit

Manage topics

2 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

norahsakal Working app — Latest commit b12a945 3 minutes ago

frontend	Working app	3 minutes ago
README.md	Working app	3 minutes ago
app.py	Working app	3 minutes ago
requirements.txt	Working app	3 minutes ago
train_your_model.ipynb	Working app	3 minutes ago

README.md

Guide to your own artificial intelligence app in 3 steps

1) Clone repo and install requirements from requirements.txt

Conclusions

DEMO

Thank you

Norah Klintberg Sakal

norahsakal

norahsakal

www.getshades.co

https://github.com/norahsakal/pycascades-2019-shades

norah.klintberg.sakal@gmail.com