# Mobile Nets

## Enabling Computer Vision on the Web

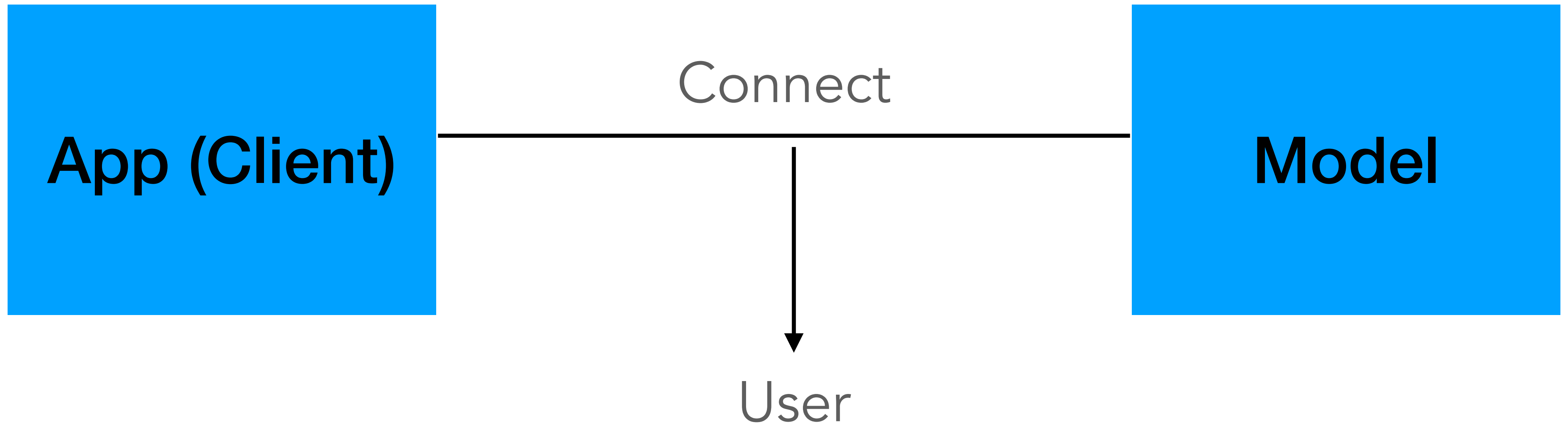
Maximilian Eissler - 09.02.2022

# Mobile Nets

## Enabling Computer Vision on the Web

Let's assume we need to serve a (CV) model

in some sort of application!




Let's assume we do image classification in a web application.

➡️ your app is going to be a Web App running on a browser (client).

Model on Server

- Inference is done on a server
- The client is thin i.e. only for I/O

Model on Client

- The server provides a model
- The client is thick, i.e., does inference as well

# Mobile Nets

## Bringing Computer Vision to the Web

### Key Questions

1. Why should I run a model on the browser?

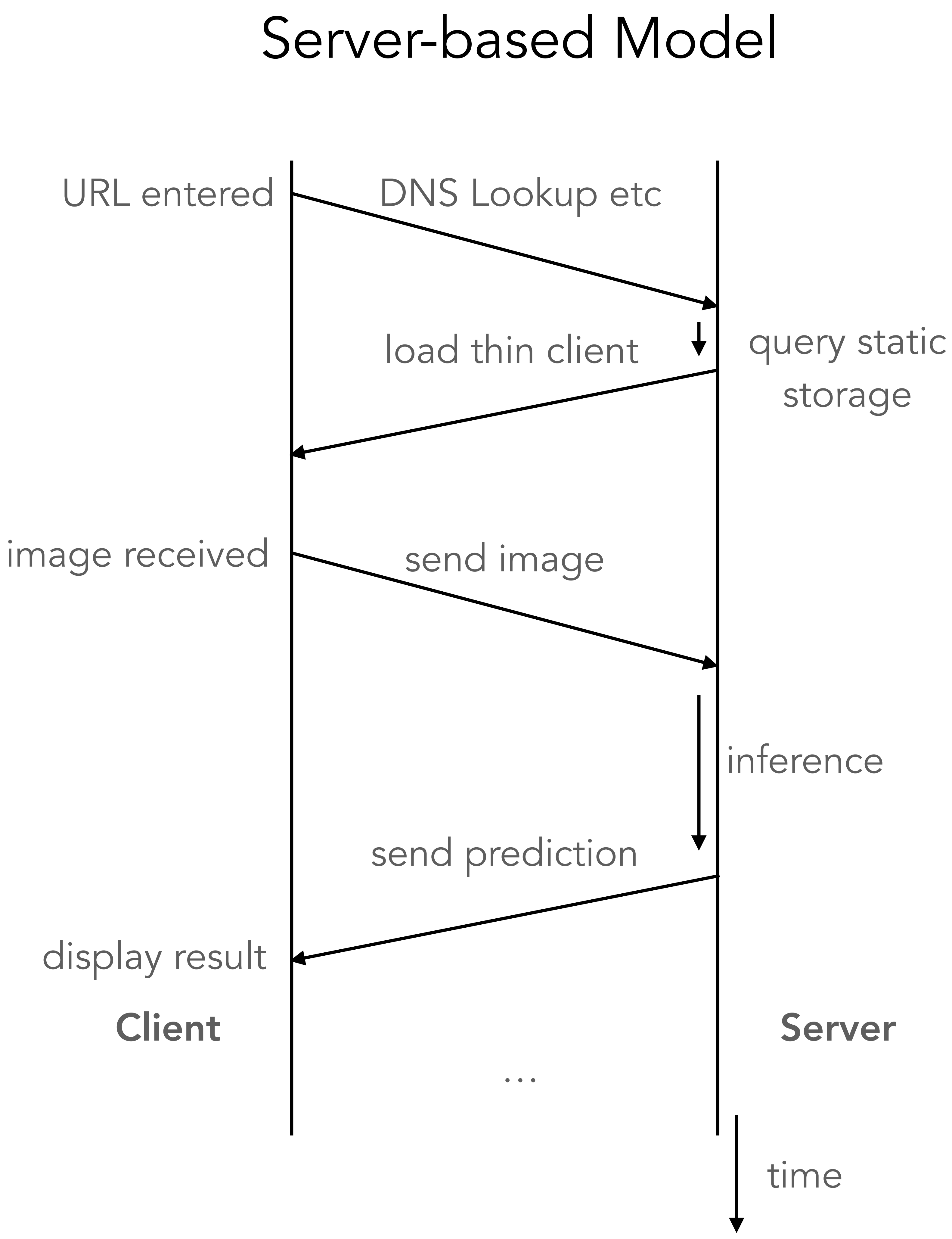
2. How do Mobile Nets work and how are they helpful?

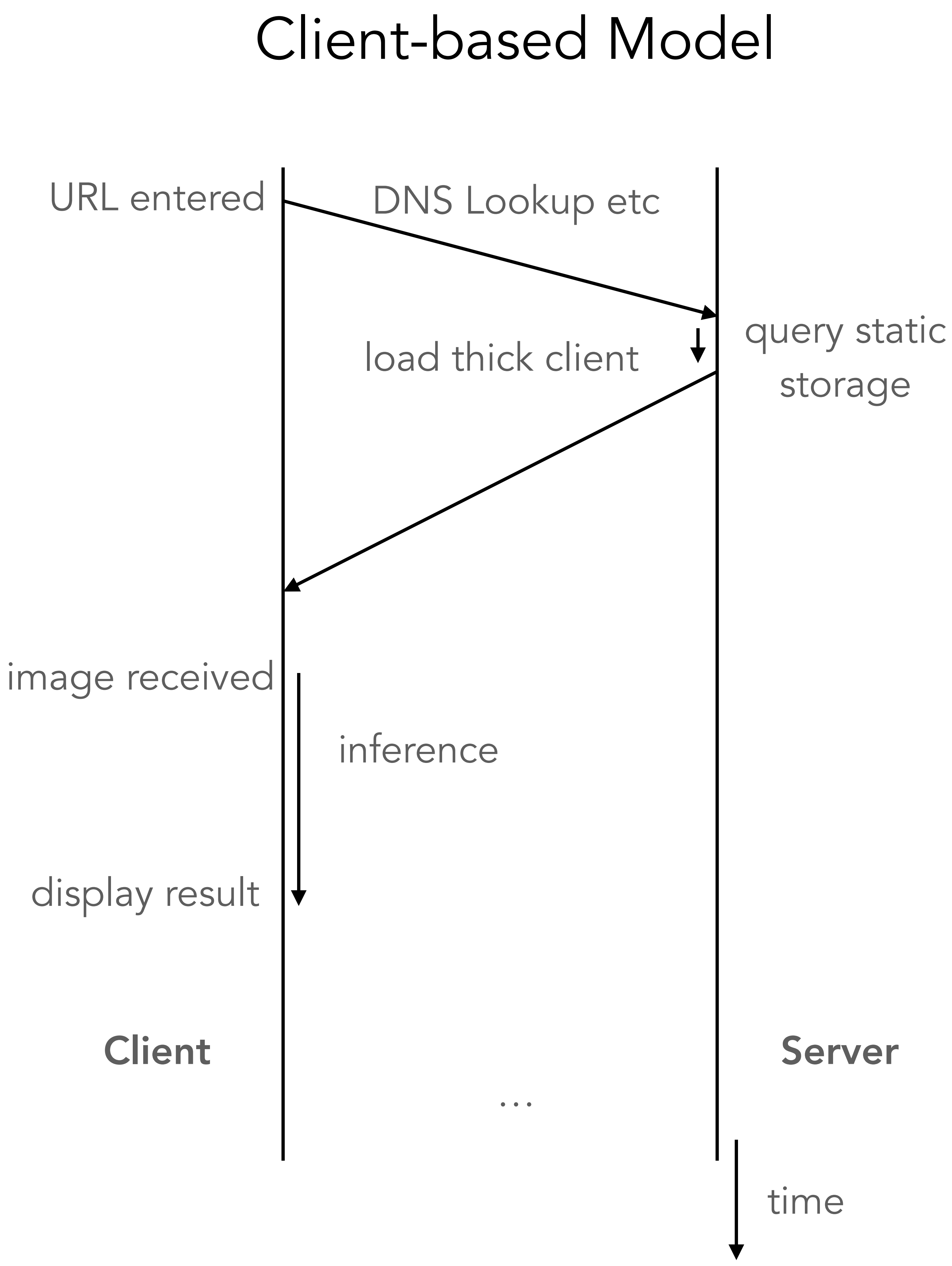
3. How **do** I run a model on the browser? (DEMO)

# Why should I run a model on the browser ?

# Why should I run a model on the browser ?

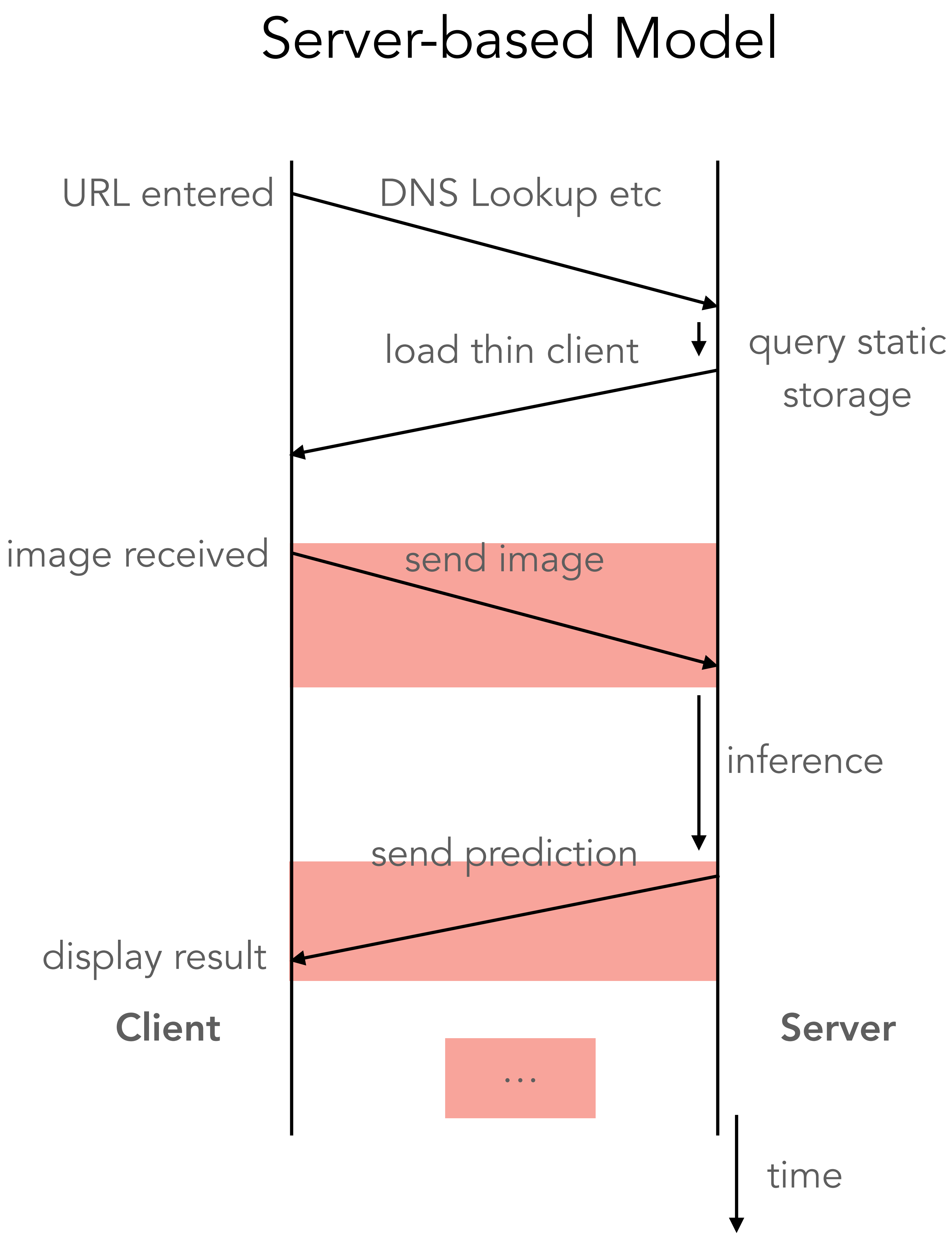
## Pros/Cons for a browser-based application

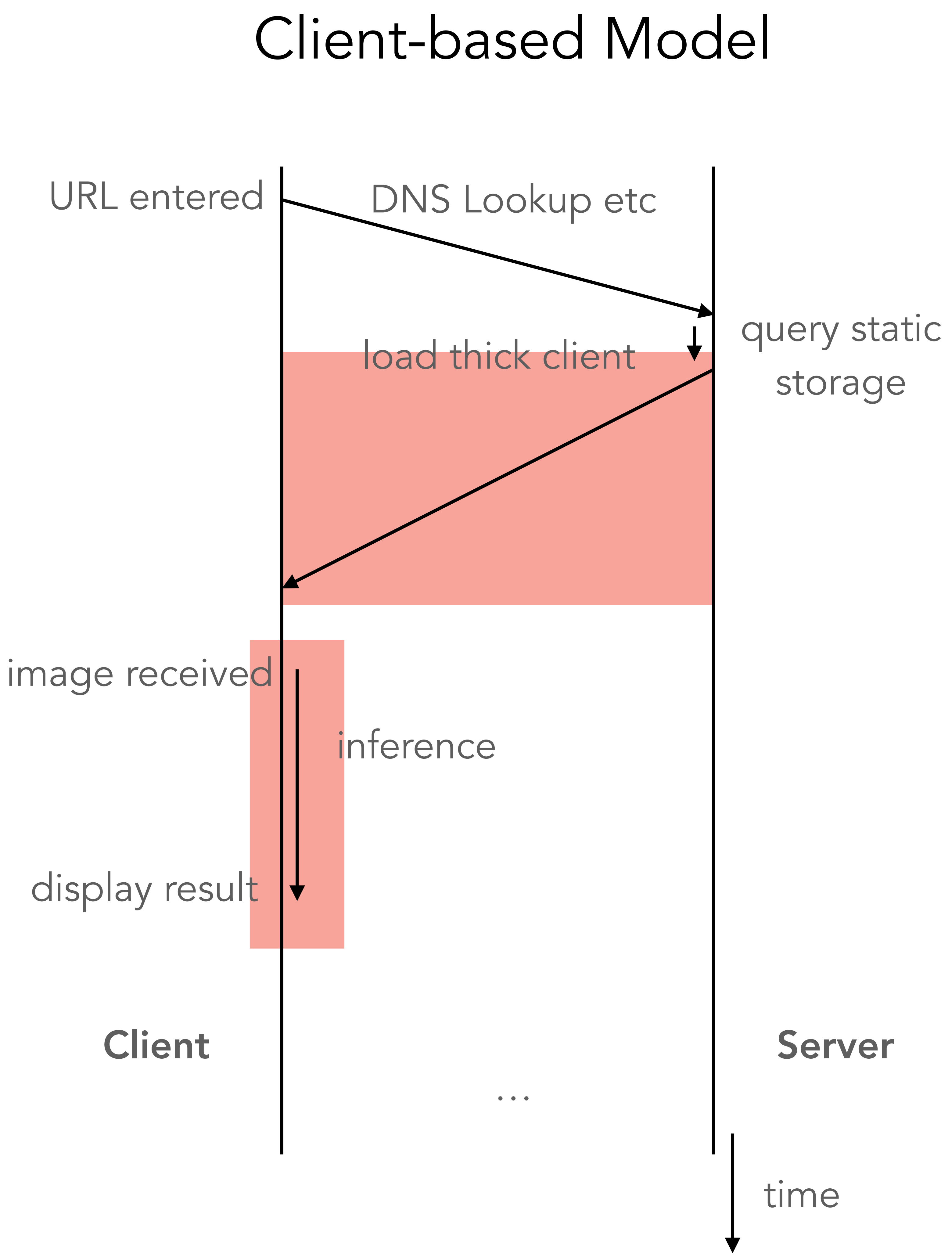


# Why should I run a model on the browser ?

## Pros/Cons for a browser-based application

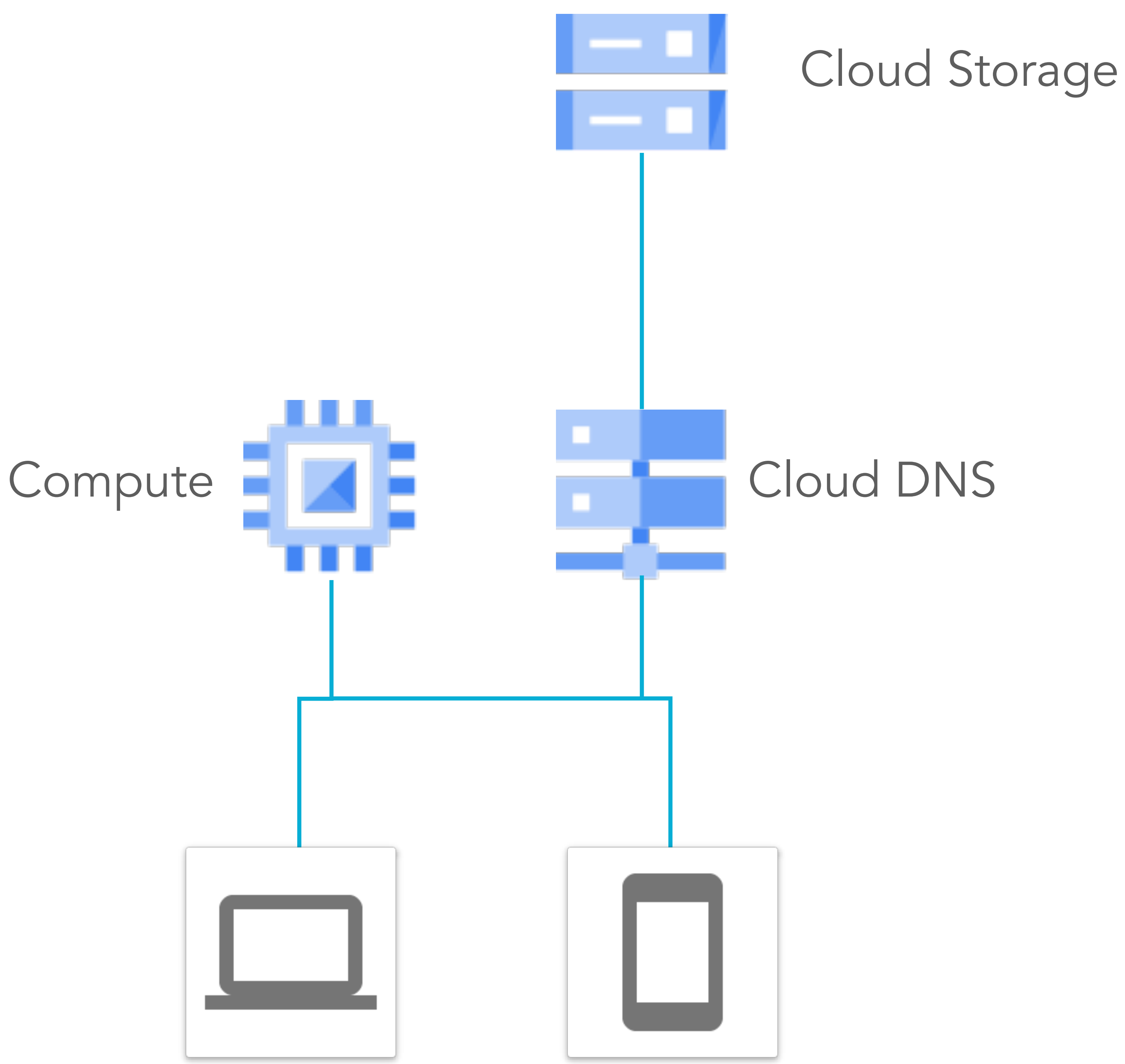


# Why should I run a model on the browser ?

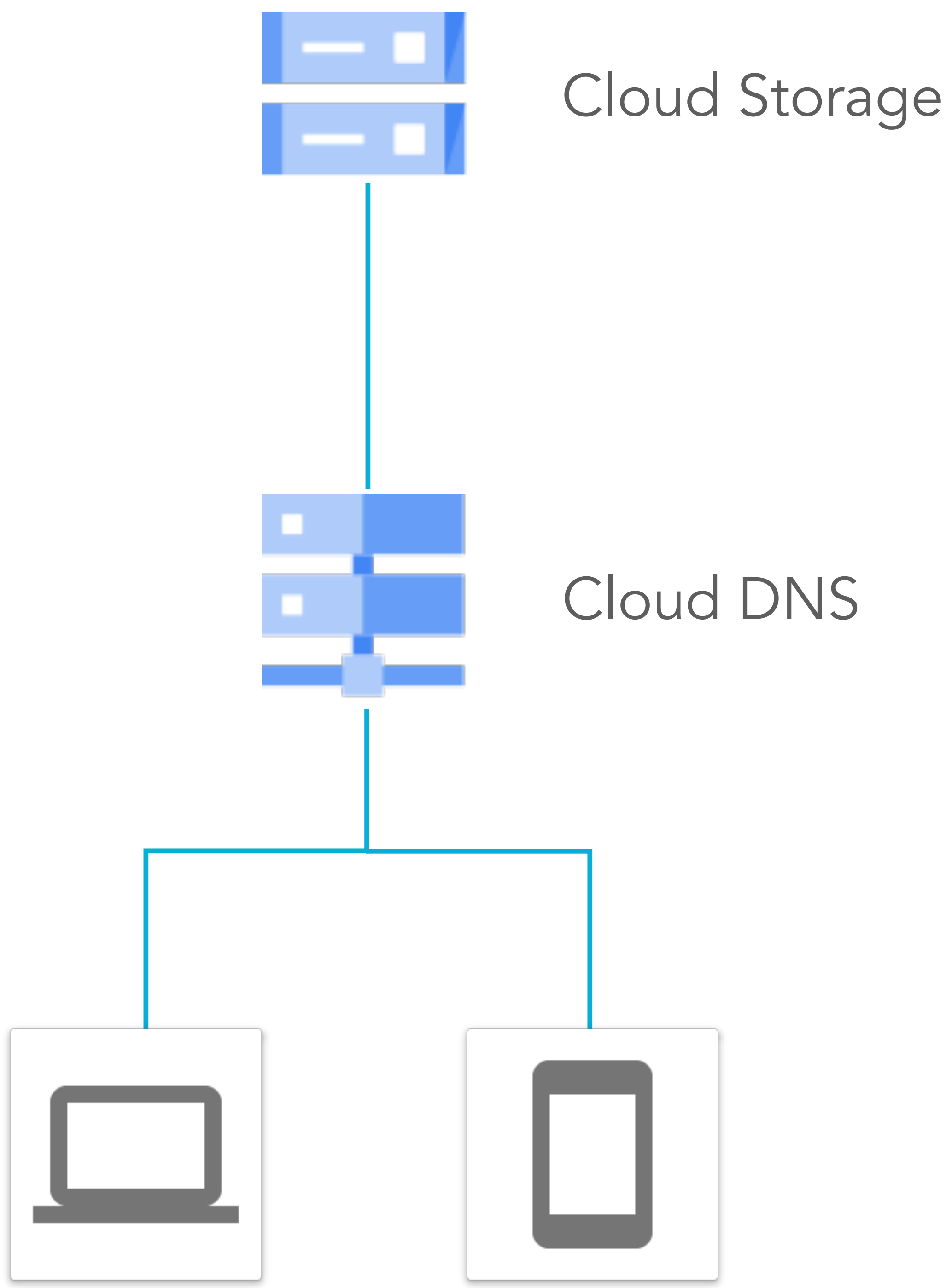
## Pros/Cons for a browser-based application

Server-based Model



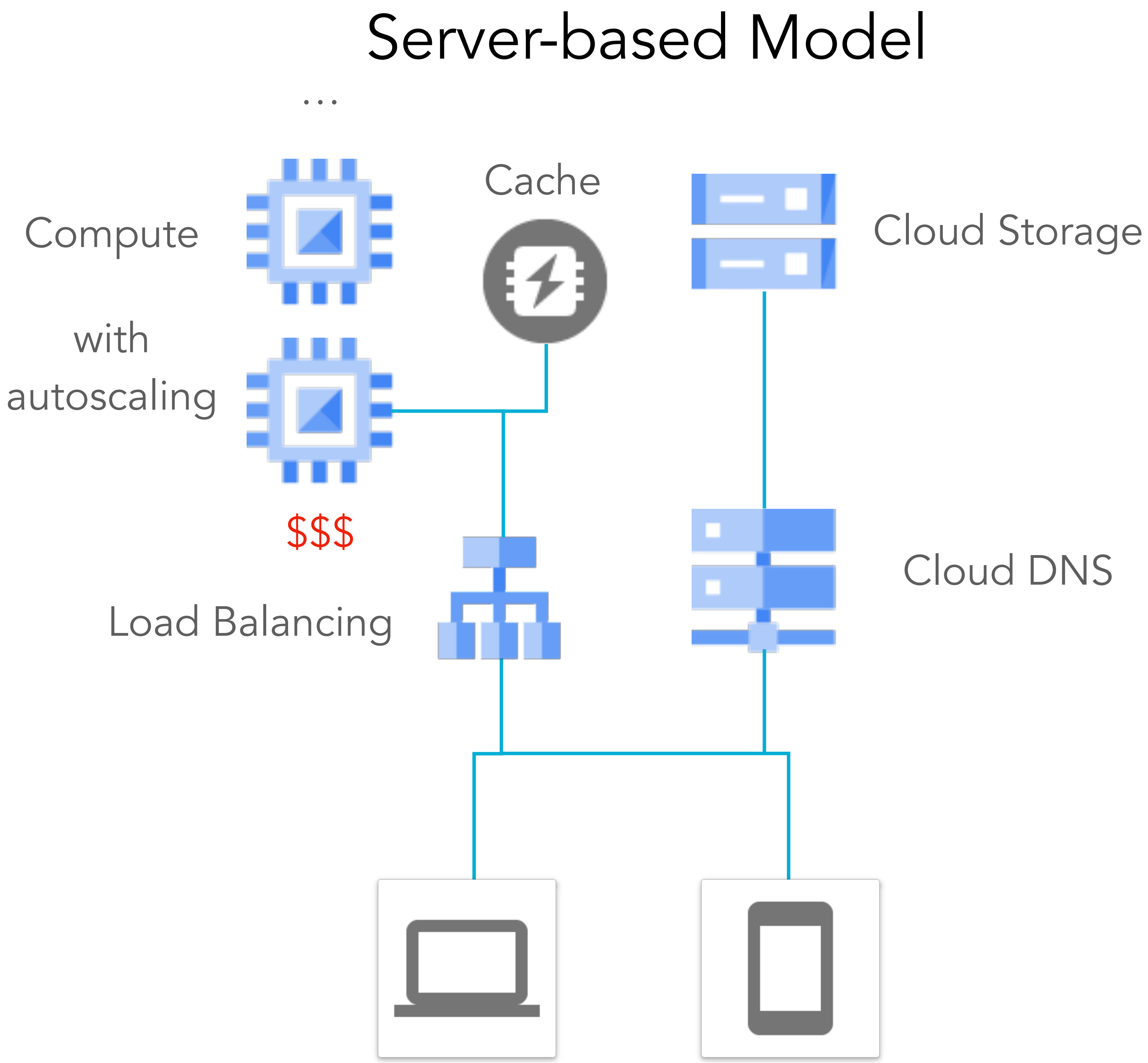

**10 Users**

Client-based Model

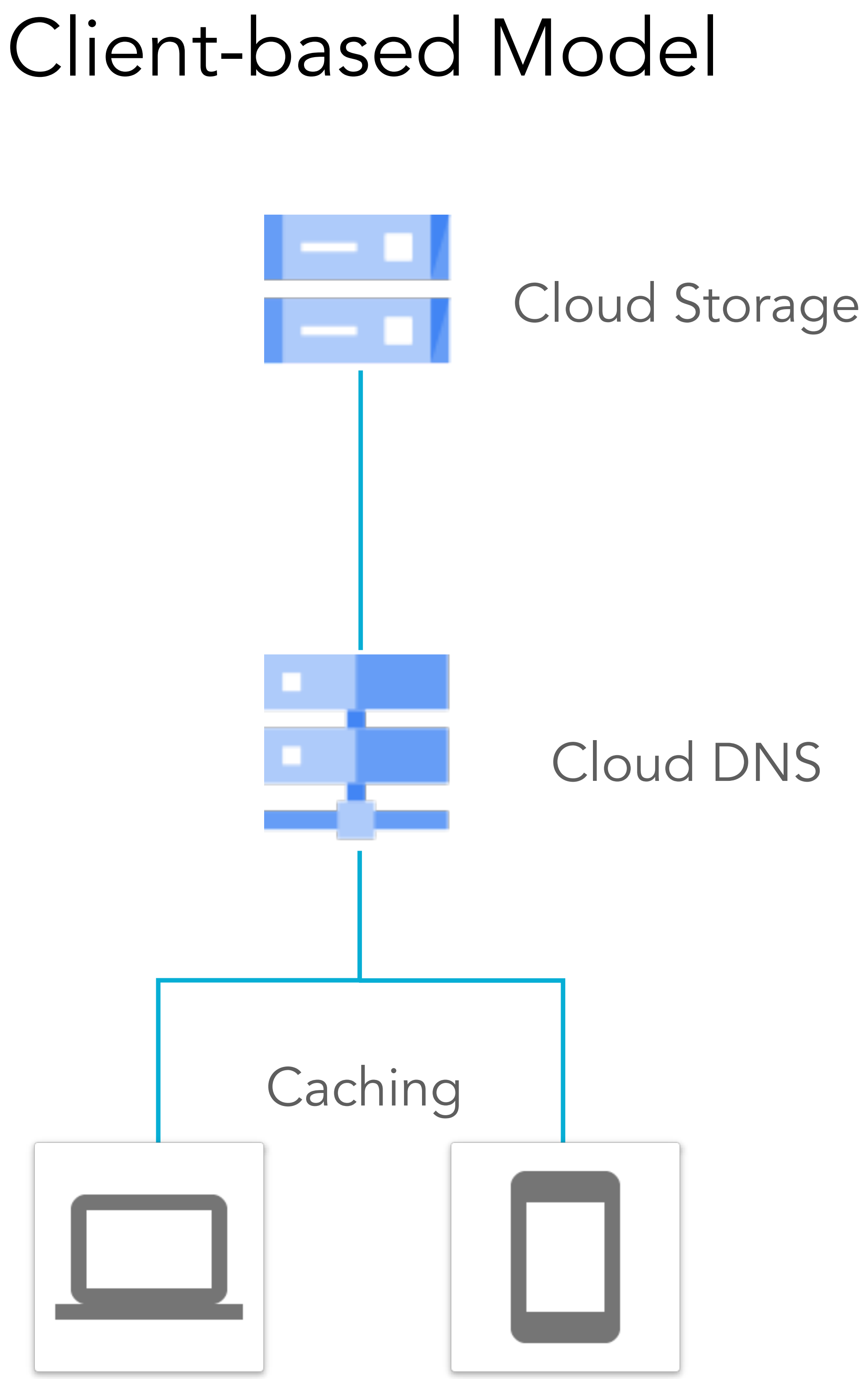


# Why should I run a model on the browser ?

## Pros/Cons for a browser-based application




**10000 Users**



# Why should I run a model on the browser ?

## Pros/Cons for a browser-based application

Server-based Model

| + | - |
|---|---|
| unlimited memory and compute** | high sustained network traffic |
| | privacy |
| | needs reliable connection |
| | cost scales with no. predictions |
| | architecture complexity scales |

Client-based Model

| + | - |
|---|---|
| low latency | slow initial load |
| privacy | memory limitations |
| cost hardly scales | compute limitations |
| simple architecture | |

# Why should I run a model on the browser ?

## Pros/Cons for a browser-based application

Server-based Model

| + | - |
|---|---|
| unlimited memory and compute** | high sustained network traffic |
| | privacy |
| | needs reliable connection |
| | cost scales with no. predictions |
| | architecture complexity scales |

Client-based Model

| + | - |
|---|---|
| low latency | slow initial load |
| privacy | memory limitations |
| cost hardly scales | compute limitations |
| simple architecture | |

# Why should I run a model on the browser ?

**If size and complexity of the model allow it, why NOT run it on the browser?**

# How do Mobile Nets work and how are they helpful?

# How do mobile nets work and how are they helpful?

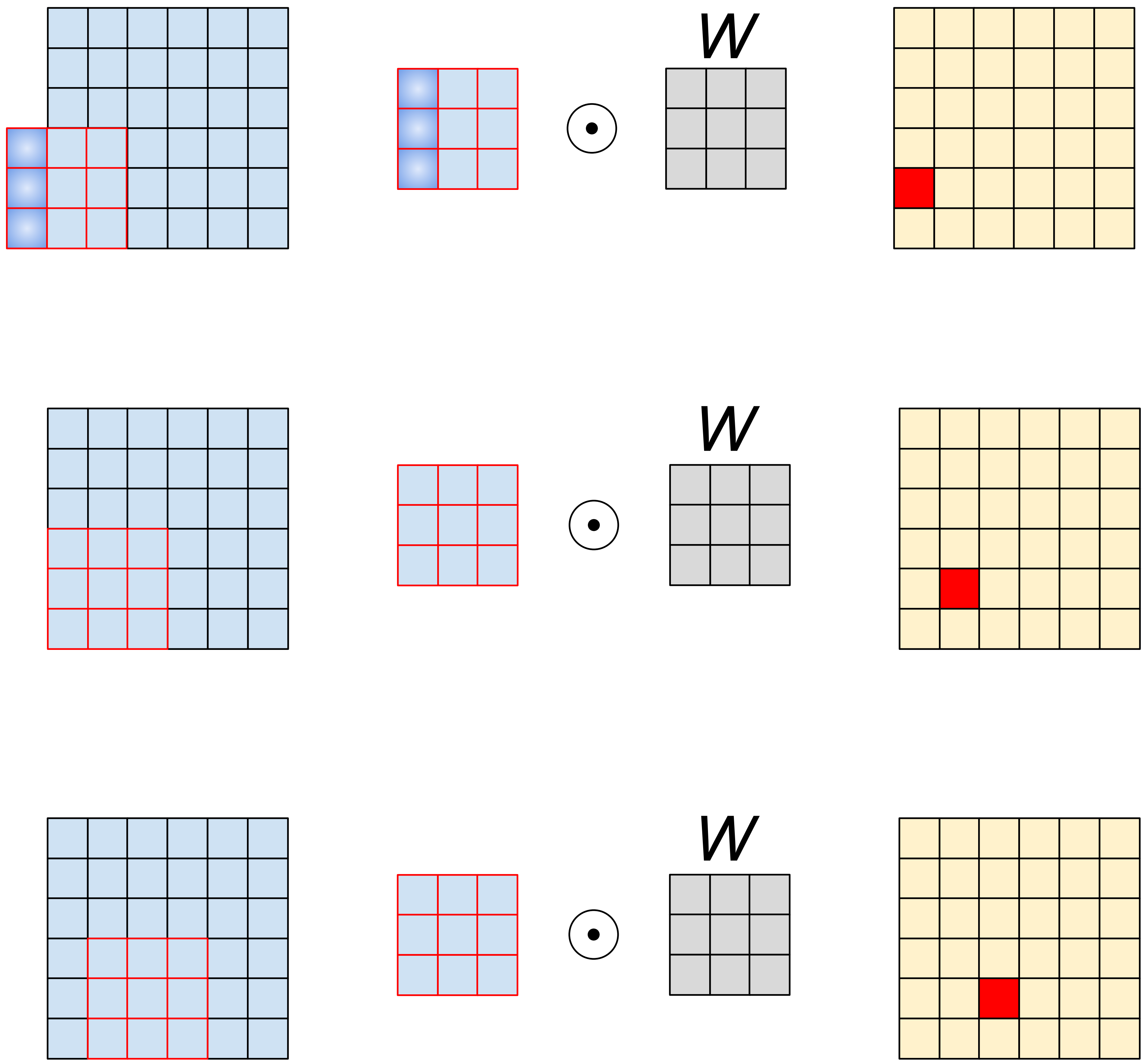
## What problem do Mobile Nets address?

- CV models are large and computationally complex
- This makes them hard to run on the client side.
- In "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" Howard et al. aim to develop SOA Classifiers which can run on the client
- They observed: Models make heavy use of convolutional operations

**Mobile Nets use depth-wise separable convolutions to reduce size & complexity**

**Mobile Nets introduce model shrinking parameters to fine-tune size & complexity**

# How do mobile nets work and how are they helpful?
## Depth-Wise Separable Convolutions

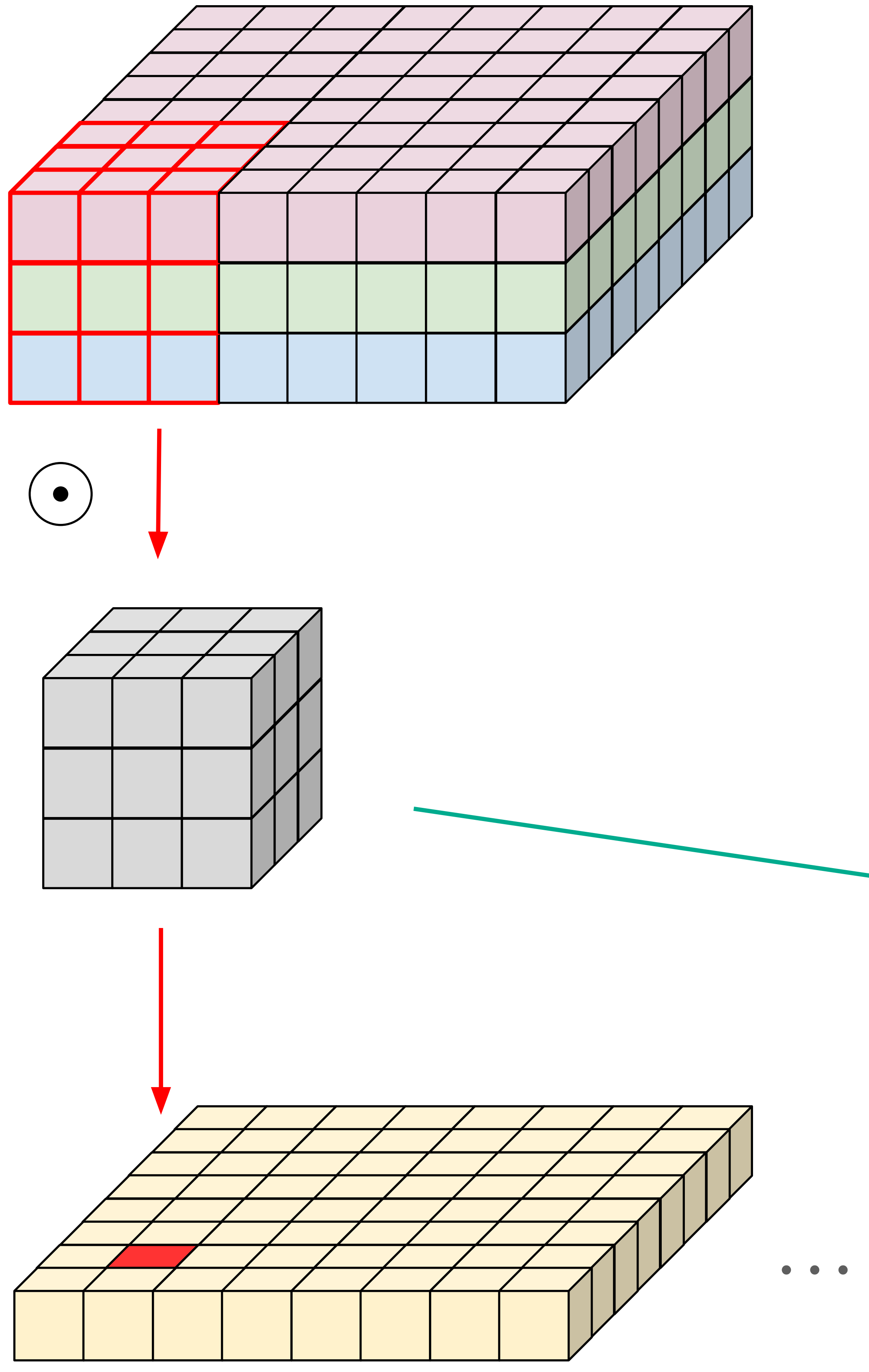



[1]

2D convolution

# How do mobile nets work and how are they helpful?

## Depth-Wise Separable Convolutions



[1]

3D convolution

Lets assume:

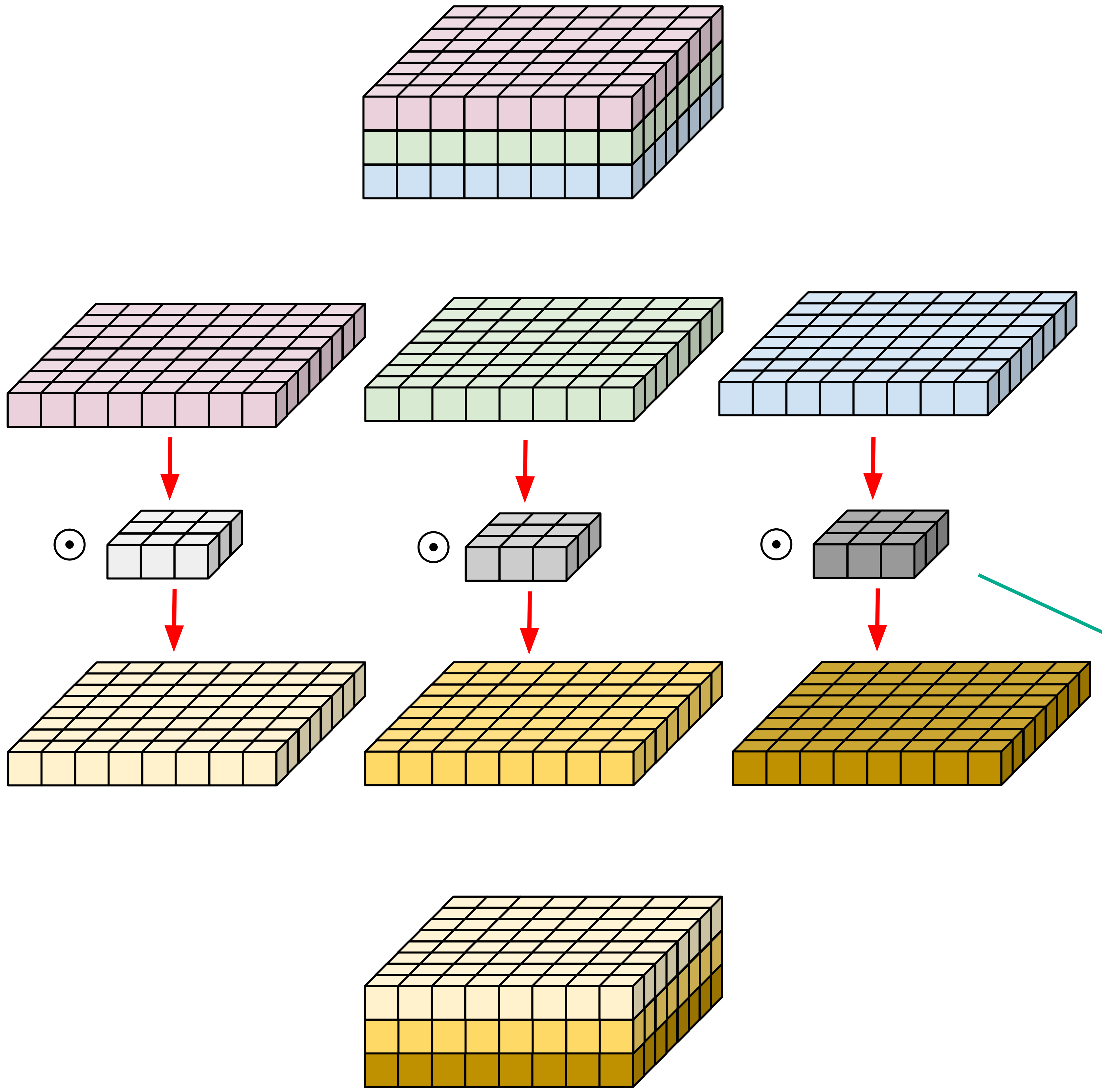
- Image size of 8x8
- padding of 1
- stride of 1
- 3 input channels
- kernel size 3x3
- N output channels

Parameters:

3 x 3 x 3 x N = 27 x N

# How do mobile nets work and how are they helpful?

## Depth-Wise Separable Convolutions




[1]

depth-wise convolution

Lets assume:

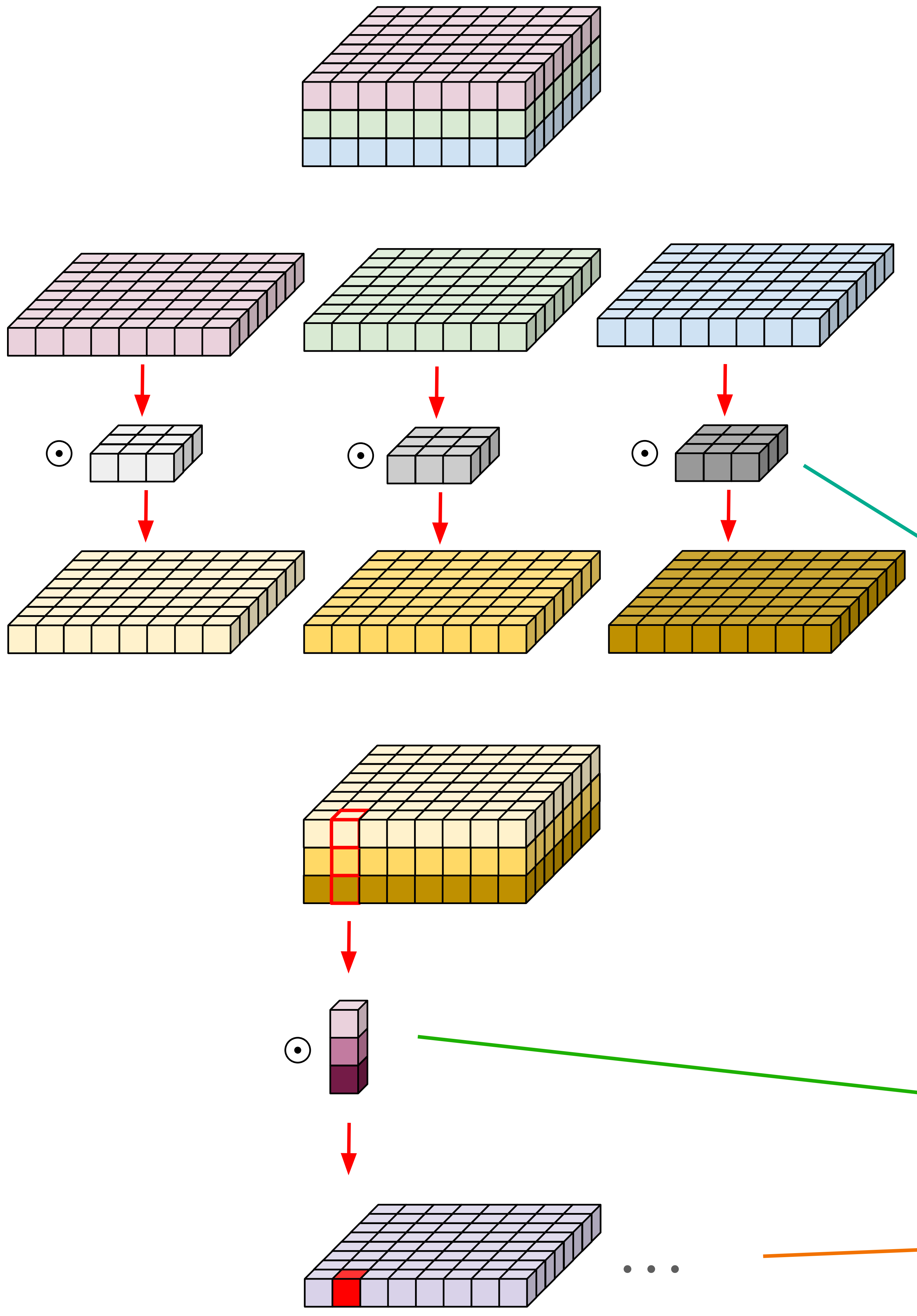
- Image size of 8x8
- padding of 1
- stride of 1
- 3 input channels
- kernel size 3x3
- N output channels

Parameters:

3 x 3 x 3 = 27

# How do mobile nets work and how are they helpful?

## Depth-Wise Separable Convolutions



[1]

depth-wise separable convolution

Lets assume:

- Image size of 8x8
- padding of 1
- stride of 1
- 3 input channels
- kernel size 3x3
- N output channels

Parameters:

27

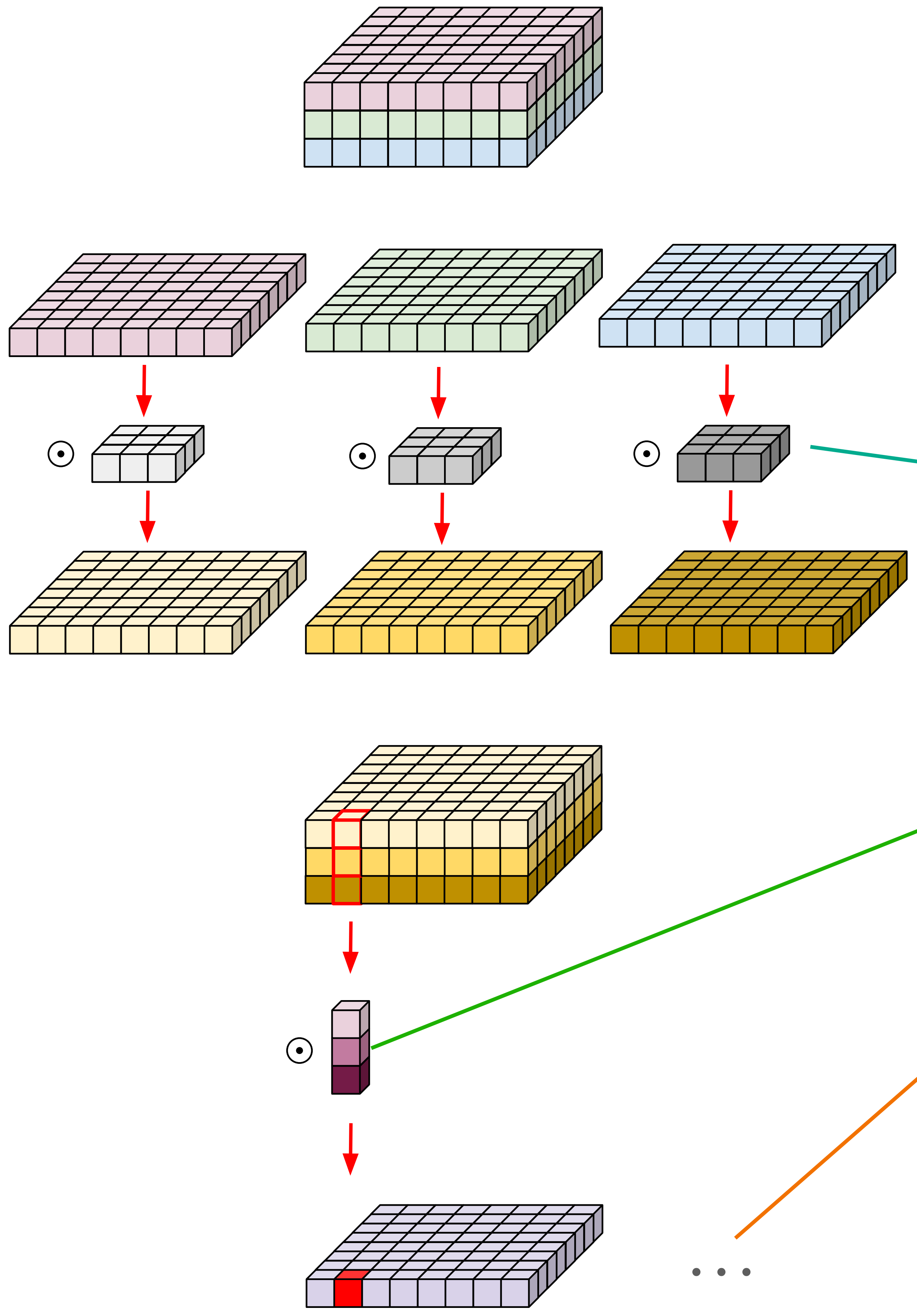
+

3 x N

# How do mobile nets work and how are they helpful?

## Depth-Wise Separable Convolutions

- 3 input channels
- kernel size 3x3
- N output channels



Parameters:

27

+

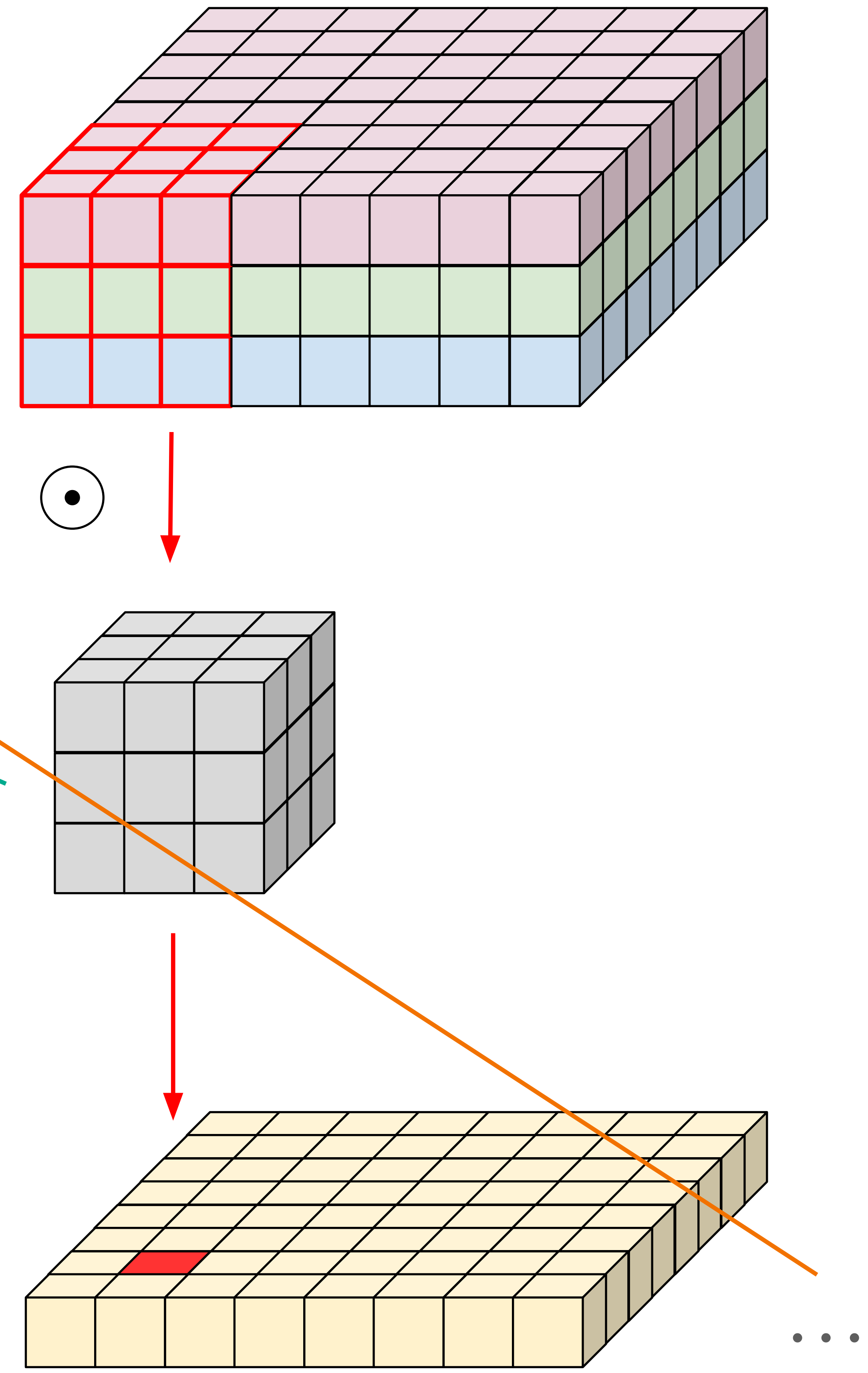
3 x N

[1]

depth-wise separable convolution

Parameters:

27 x N



[1]

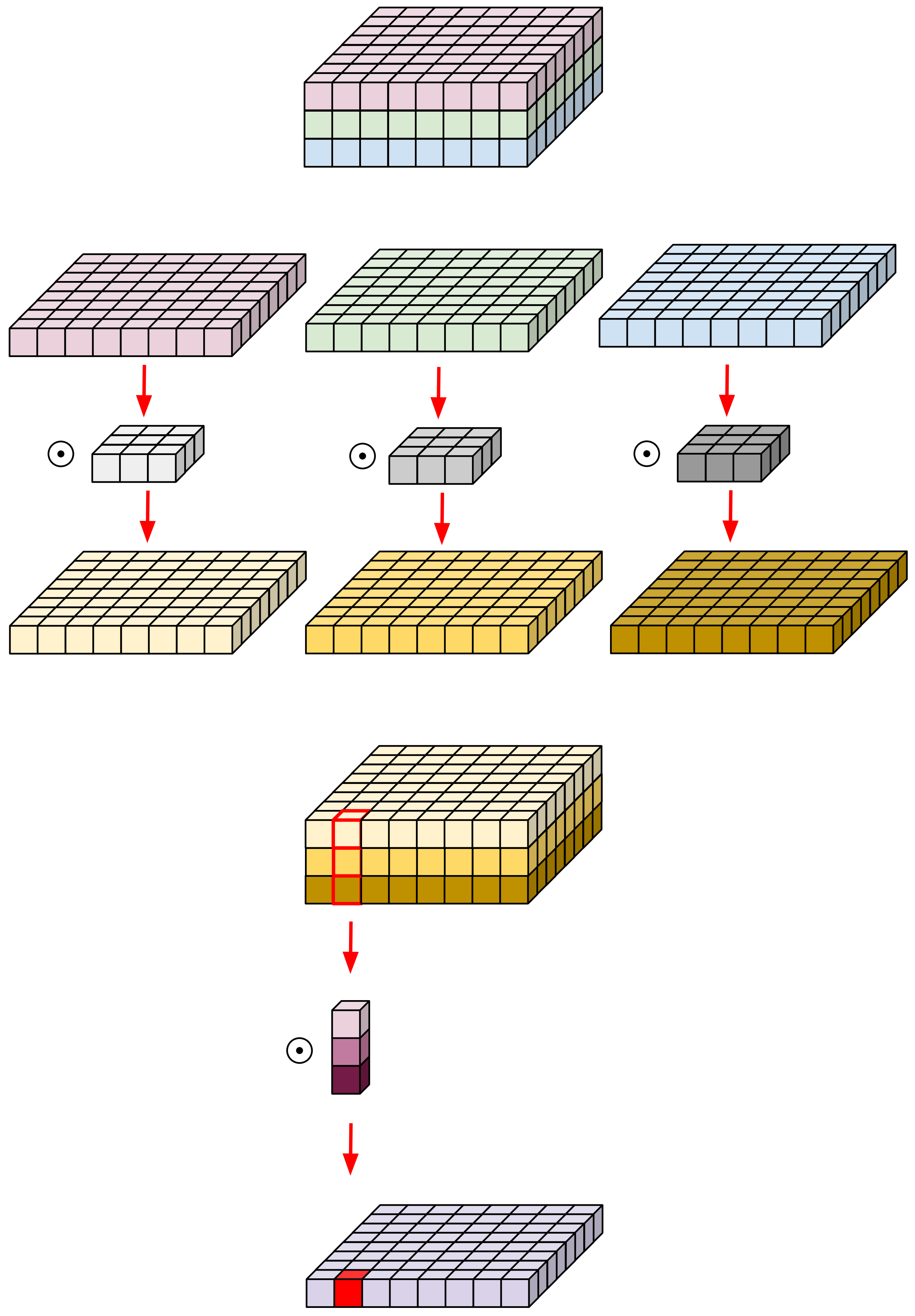
3D convolution

Advantage

$$\frac{1}{N} + \frac{1}{9}$$

# How do mobile nets work and how are they helpful?

## Depth-Wise Separable Convolutions



[1]

depth-wise separable convolution

Limitation:

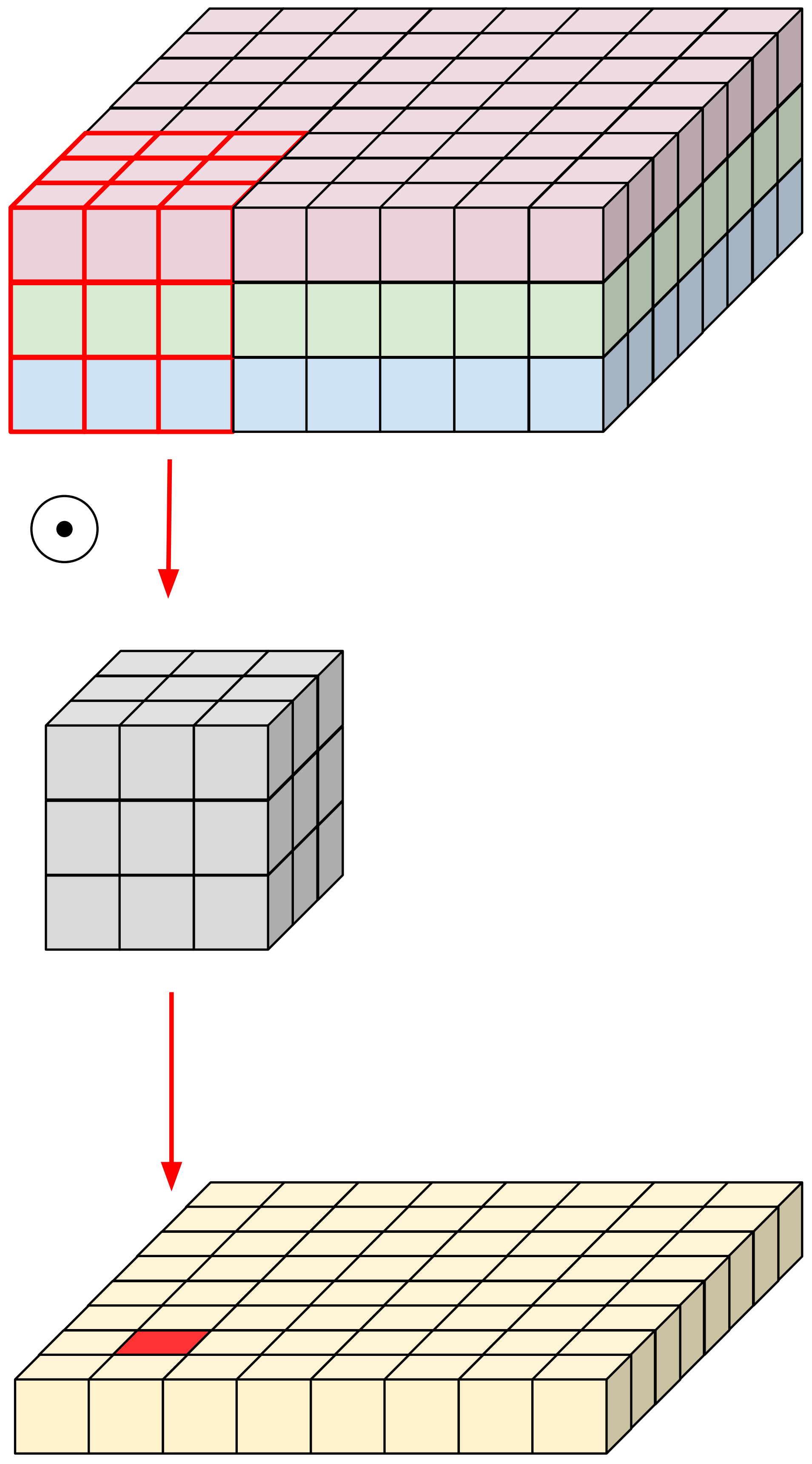
Our filters cannot mix input channels!

It seems to have little effect in practice:

Table 4. Depthwise Separable vs Full Convolution MobileNet

| Model | ImageNet Accuracy | Million Mult-Adds | Million Parameters |
|---|---|---|---|
| Conv MobileNet | 71.7% | 4866 | 29.3 |
| MobileNet | 70.6% | 569 | 4.2 |

[2]



[1]

3D convolution

# How do mobile nets work and how are they helpful?

## Model Scaling Parameters

- In addition to depth-wise separable convolutions, two model scaling parameters are introduced.
- α, set in (0,1] scales the number of input and output channels in each layer.
- Reduces the number of parameters and reduces computational cost by roughly $\alpha^2$
- ρ set in (0,1], scales the resolution of the images and subsequently the resolution of each internal representation (between the layers).
- Reduces computational cost by $\rho^2$

| Layer/Modification | Million Mult-Adds | Million Parameters |
|---|---|---|
| Convolution | 462 | 2.36 |
| Depthwise Separable Conv | 52.3 | 0.27 |
| $\alpha = 0.75$ | 29.6 | 0.15 |
| $\rho = 0.714$ | 15.1 | 0.15 |

[2]

# How do Mobile Nets work and how are they helpful?

**Mobile nets are competitive, efficient CNNs, which can easily be scaled down in terms of size and complexity.**

# How do I run a model on the browser?

DEMO

# Any Questions?

**mx-e/cv-seminar-deployment-showcase**

# Sources


- [1] A really great blog post about depth-separable convolutions:
  https://eli.thegreenplace.net/2018/depthwise-separable-convolutions-for-machine-learning/
- [2] The Mobile Nets paper:
  https://arxiv.org/pdf/1704.04861.pdf

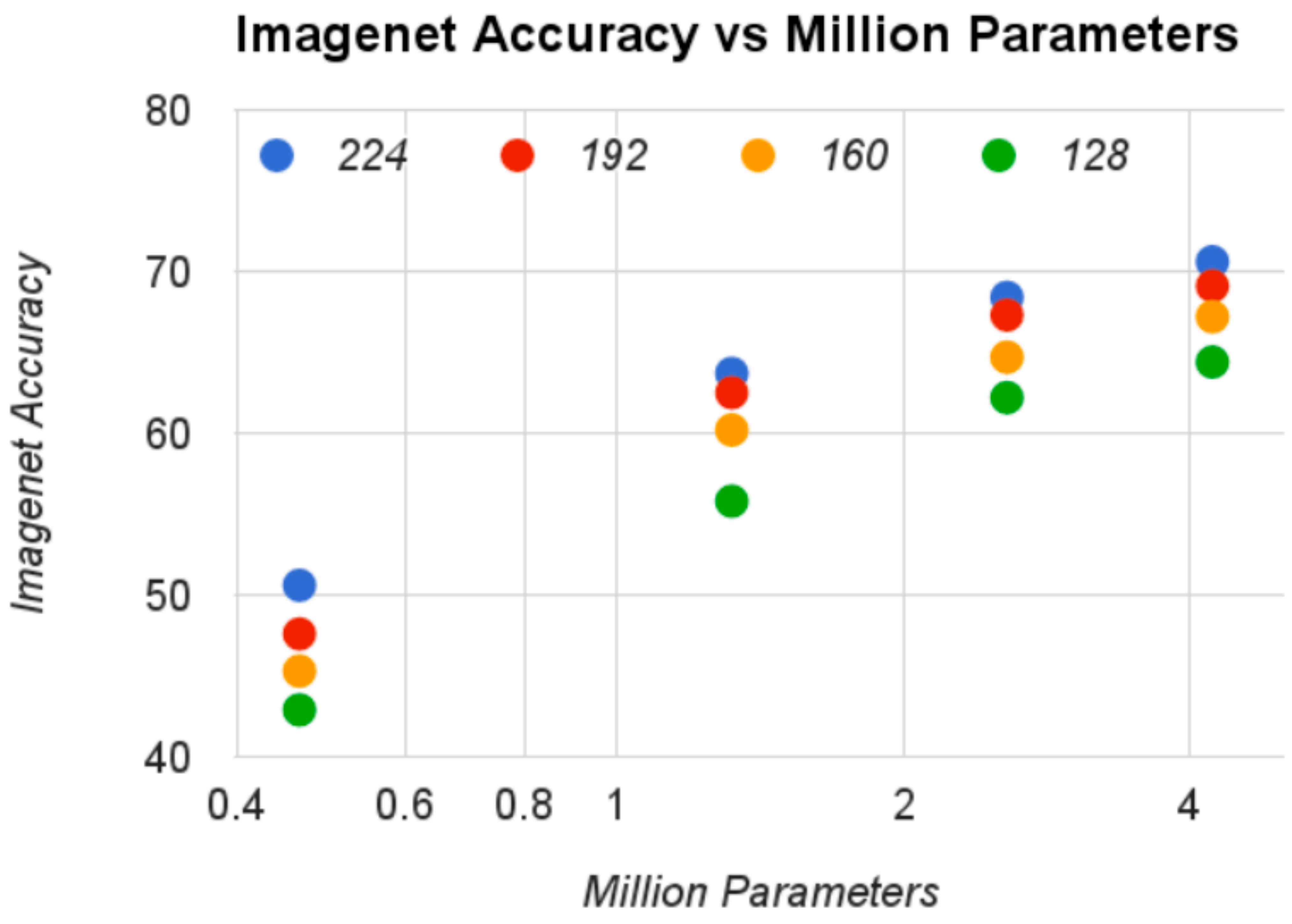

Further Reading:

- A basic tutorial on how to set up an inference backend using python, FastAPI and Tensorflow:
  https://towardsdatascience.com/image-classification-api-with-tensorflow-and-fastapi-fc85dc6d39e8
- A tutorial on handling image uploads in react via drag and drop:
  https://medium.com/@650egor/simple-drag-and-drop-file-upload-in-react-2cb409d88929
- A tutorial on using TFJS with react:
  https://levelup.gitconnected.com/build-ad-dog-classifier-with-react-and-tensorflow-js-in-minutes-f08e98608a65

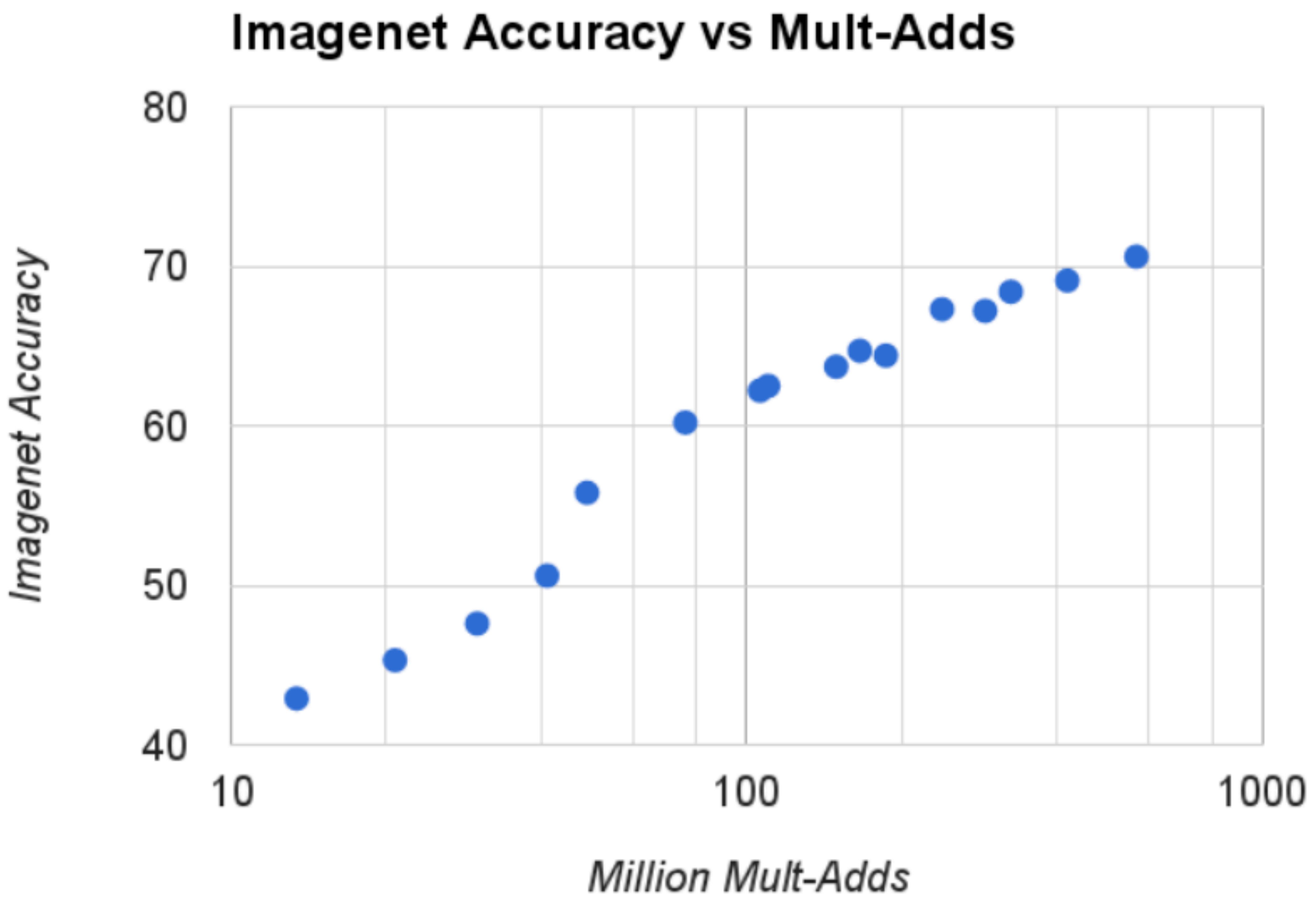
# Additional Slides

# How do mobile nets work and how are they helpful?

## Model Scaling Parameters




[2]




[2]

# How do I run a model on the browser?

## Overview

### Server-based Model

**Thin Client**

**Upload files here!**

| llama | Egyptian_cat | banana |
|---|---|---|
| 58.80 % | 83.68 % | 99.74 % |
| hog | lynx | hook |
| 24.43 % | 5.88 % | 0.06 % |

### Client-based Model

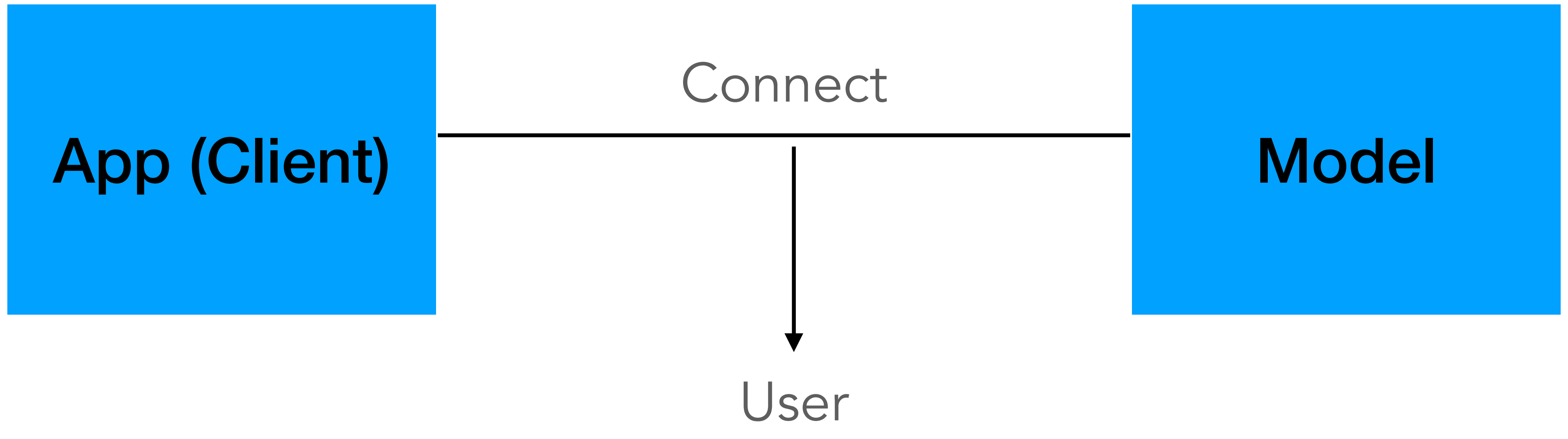
**Thick Client**

**Upload files here!**

| llama | tiger cat | banana |
|---|---|---|
| 0.29 | 0.33 | 0.84 |
| kelpie | Persian cat | nipple |
| 0.11 | 0.27 | 0.14 |
| Eskimo dog, husky | tabby, tabby cat | spaghetti squash |
| 0.1 | 0.14 | 0.01 |

# How do I run a model on the browser?

## Overview




1. Write Application (I/O, Presentation, etc.)

2. Implement Model (Backend, Parameter Loading, Inference, etc.) on the client

3. Make the app available to the user

# How do I run a model on the browser?

## Overview

1

Write Application
(I/O, Presentation, etc.)

2

Implement Model
(Parameter Loading,
Inference, etc.)
on the client

3

Make the app available
to the user

# How do I run a model on the browser?

## Writing a browser application

- The app has to be able to upload dropped images and display them.
- There are many viable options to build an application in the browser (for example React, Vue, Angular, native Javascript + HTML + CSS, Typescript, etc…)
- I used React.js to write the UI.

```
import { useState } from "react";
import { Upload } from "./upload";
import ImageViewer from "./image-viewer";

const useImgs = (initialState : any[] = [], maxFiles : number = 3) => {
  const [state, setstate] = useState(initialState);
  const addImgs = (newDrops) => {
    const newImgs = newDrops
      .map((file) => {
        if (file.type.includes( value: "image")) {
          file.preview = URL.createObjectURL(file);
          return file;
        }
        return null;
      })
      .filter((elem) => elem !== null);

    setstate([...newImgs, ...state].slice(0, maxFiles));
  };
  return [state, addImgs];
};

const ImageClassifier = () => {
  const [imgs, addImgs] = useImgs( initialState: []);
  const onFileDrop = (files) => {
    console.log(files);
    if (files.length > 0) {
      addImgs([...files]);
    }
  };
  return (
    <div>
      <h2>Thin Client</h2>
      <Upload onDrop={onFileDrop} />
      <ImageViewer files={imgs} />
    </div>
  );
};

export default ImageClassifier;
```

# How do I run a model on the browser?

## Integrating a model

- TFJS(Tensorflow JS)
- "TensorFlow.js is an open-source hardware-accelerated JavaScript library for training and deploying machine learning models."
  https://github.com/tensorflow/tfjs
- To run a mobile net in tfjs requires 2 lines of code

```
import * as mobilenet from "@tensorflow-models/mobilenet";
import * as tf from "@tensorflow/tfjs";
```

```
const App = () => {
  const [model, setModel] = useState( initialState: null);
  useEffect( effect: () => {
    const getModel = async () => {
      const m = await mobilenet.load();
      setModel(m);
    };
    getModel();
  }, deps: []);
```

```
const PredictionDisplay = ({ model, img }) => {
  const [predictions, setPrediction] = useState( initialState: null);
  useEffect( effect: () => {
    const fetchClassification = async () => {
      const pred = await model.classify(img);
      setPrediction(pred);
    };
    setPrediction( value: null);
    fetchClassification();
  }, deps: [img]);

  return (
    <PredictionList>
```

# How do I run a model on the browser?

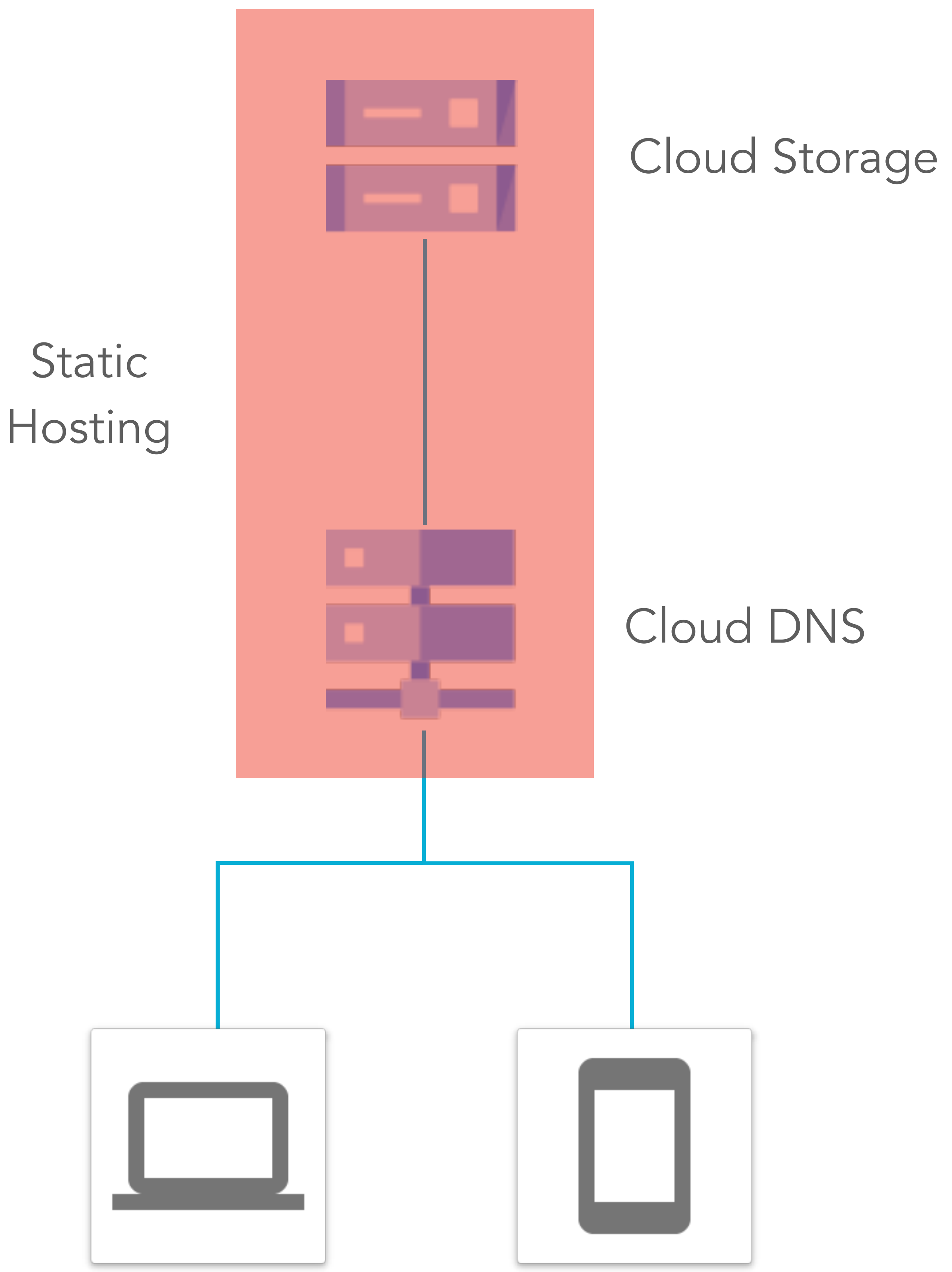
## Serving a browser CV application

- Since all dynamic data (i.e. the images to classify) remain at the client, only static data needs to be served.
- There are lots of Cloud Providers with really affordable (or free) offers for static hosting
- Examples include Github Pages, Firebase Hosting, …

GitHub Pages

Firebase Hosting

HEROKU



# How do I run a model on the browser?

## What if I want to use a thin client?

```javascript
import axios from "axios"

const request = axios.create({
  baseURL: 'http://localhost:8000',
  timeout: 10000,
});

const getPrediction = async (image) => {
  let blob = await fetch(image.preview).then(r => r.blob());
  const formData = new FormData();
  formData.append( name: 'file', blob);
  return request.post( url: '/predict/image', formData,
     config: {headers: { "Content-Type": "multipart/form-data" }})
}

export {getPrediction}
```

We need to add a request middleware to the web app …

```python
import uvicorn
from fastapi import FastAPI, File, UploadFile
from serve_model import predict, read_imagefile


app = FastAPI()
@app.post("/predict/image")
async def predict_api(file: UploadFile = File(...)):
    print(file.filename)
    image = read_imagefile(await file.read())
    prediction = predict(image)
    return prediction


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

…,which calls the API endpoint of a python-based server.