Esettanulmány

egy komplex képi klasszifikációs feladat hatékony megoldása felhő környezetben

Kertész Gábor

kertesz.gabor@sztaki.hu

A feladat

- Self-checkout, avagy nincs szükség pénztárosra

A feladat

- Vonalkód hiányában nem alkalmazhatóak

Viscovery

- A Viscovery egy tajvani cég, amely azzal lett ismert, hogy Japánban nyitottak egy önkiszolgáló pékséget

Bakery Visual Checkout

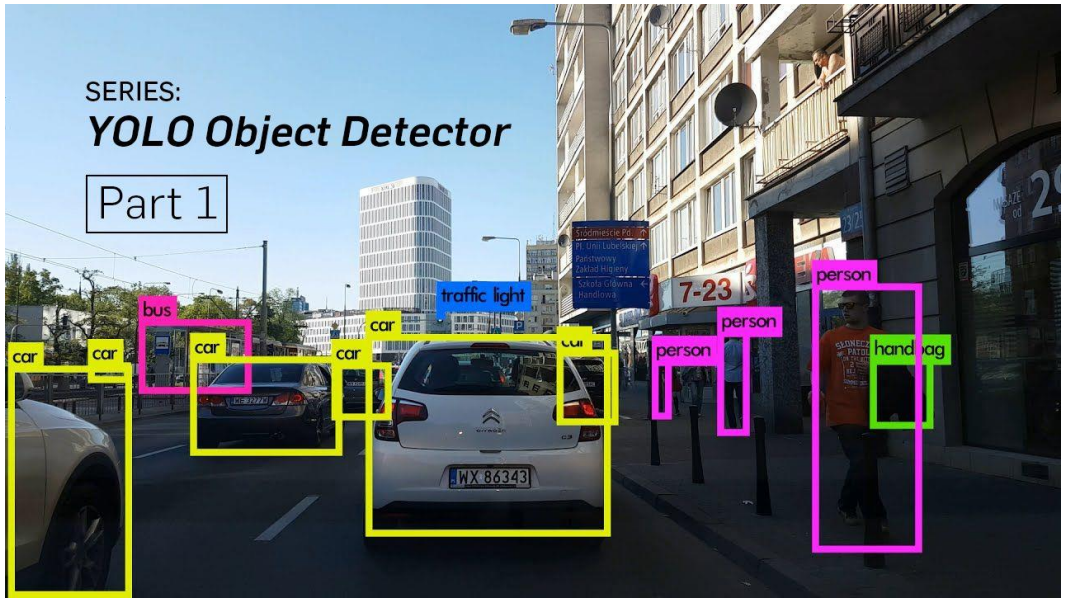
- A termékek a képük alapján kerülnek azonosításra és kiszámlázásra
- Egy képen több termék is

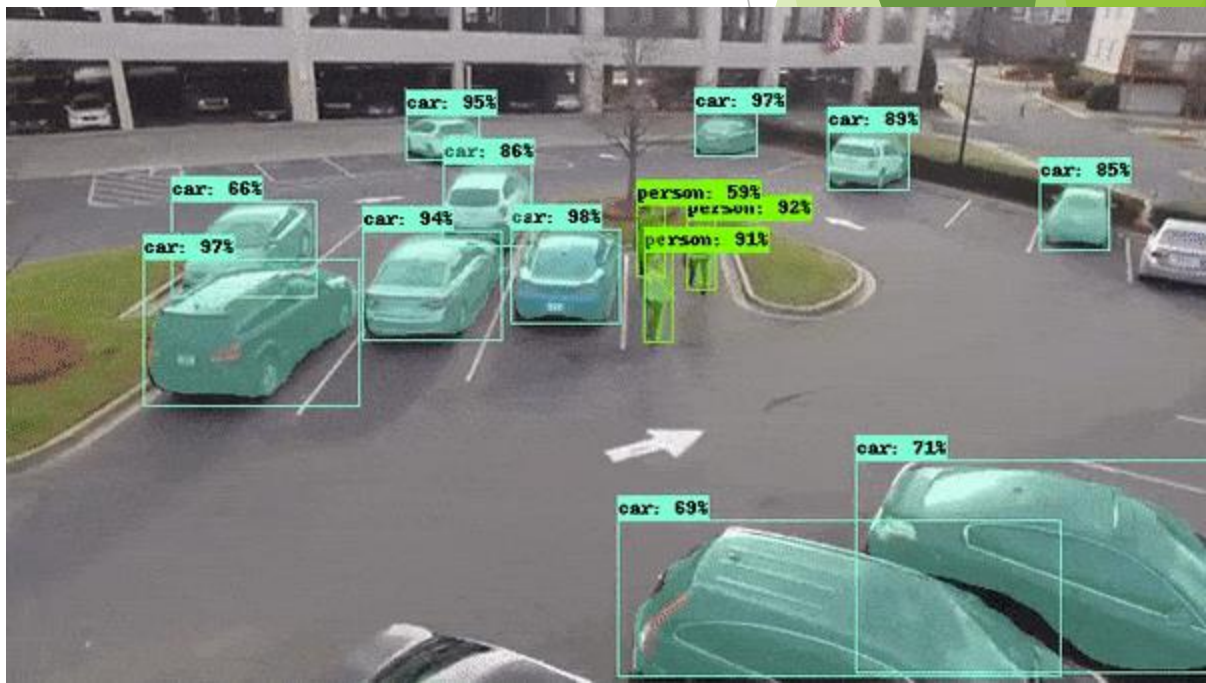
„LPDS Café”

- Egy hasonló, de kicsivel egyszerűbb problémát oldunk meg
- Egy képet szolgáltatunk a modellnek, amely feladata eldönteni, hogy mi van a képen:
  - Sajttorta
  - Fánk
  - Jégkrém
  - Macaron
  - Palacsinta

Megjegyzés: egy képen több objektum?

- Rendelkezésre álló adatok hiányában mi ettől eltekintünk, de a probléma jól ismert, és megoldható
- Objektumok detektálása, lokalizálása és klasszifikálása
  - R-CNN: Regions with CNN features
  - YOLO: You Only Look Once





Adatok

- Food-101 dataset: https://www.vision.ee.ethz.ch/datasets_extra/food-101/
- Lukas Bossard, Matthieu Guillaumin, Luc Van Gool: Food-101 – Mining Discriminative Components with Random Forests, ECCV2014
- 101 kategória, kategóriánként 1000 képpel
  - Ebből 250 kézzel válogatott, a maradék 750 az egykori **FoodSpotting** appról származott

Adatok

Adatok

- Különböző méretűek
- Különböző színek
- Különféle felbontás, minőség
- Más-más szögből fotózva

- Nem lenne könnyű algoritmizálni a kategorizálást

Gépi tanulás

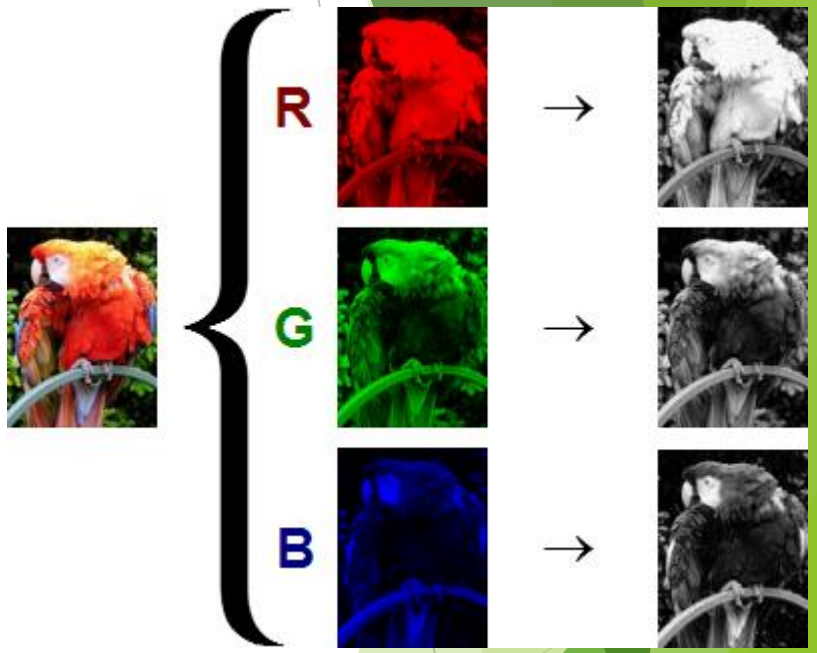
- Felügyelt tanulás
- Mutassunk sok példát a modellnek az egyes kategóriákból
  - Tehát tanítsuk be ezek felismerésére
- Majd teszteljük a klasszifikáció pontosságát!

Lépések

1. Képek kezelése
   - Tanítás és validálás
2. Hálózati architektúra kidolgozása
   - Többosztályú klasszifikáció
3. Tanítás
   - Nyomonkövetés

Képek kezelése

- A hálózat bemeneteként egy 2D képet fogunk átadni
  - Mivel színes kép, ezért a mátrix 3D:
    - Szélesség × Magasság × 3
  - Külön réteg az R, G és B csatornáknak
  - A modell feladata a színtérben levő jellemzők felismerése
- Mekkora legyen ez a kép?
  - Ha nagy, akkor jelentős a memóriaigény, lassú a tanítás
  - Ha kicsi, akkor kisebb memóriaköltség, gyors tanítás, de a tömörítés hatására romlik a diszkriminativitás

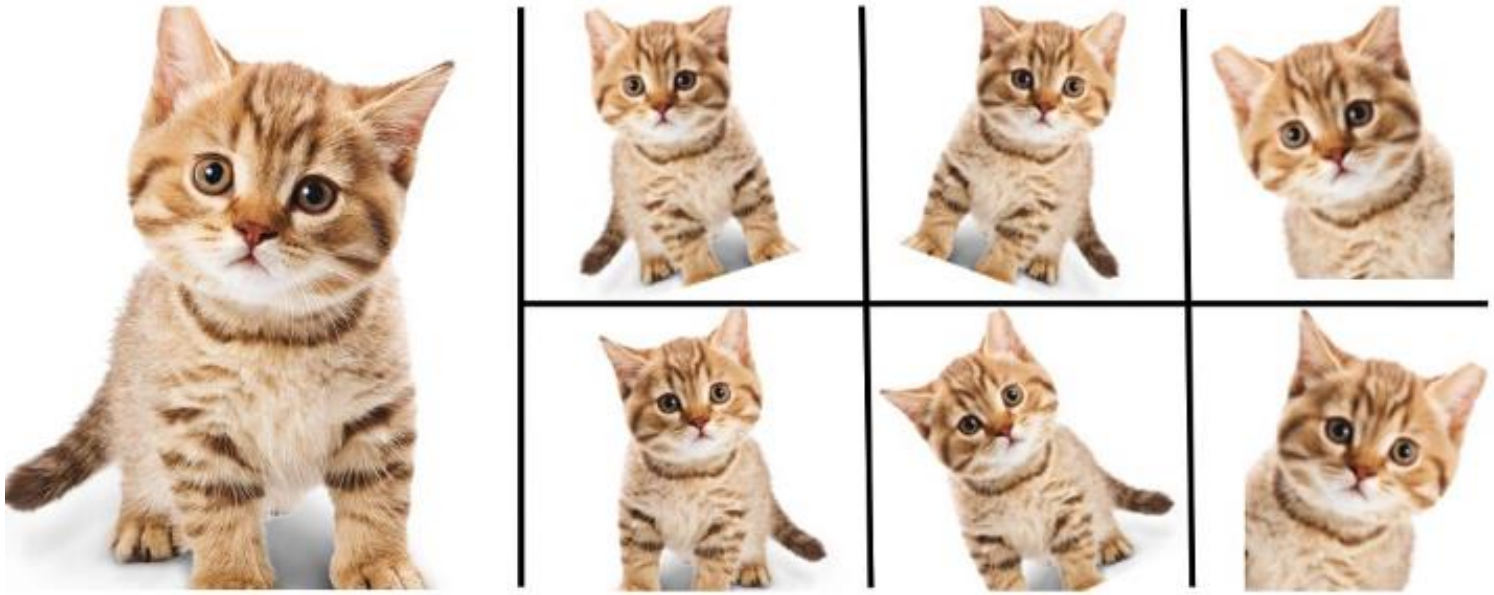


Képek átméretezésekor

5%

Data Augmentation

- A tanítóminták számát „növelhetjük” a képek manipulálásával
  - forgatásokkal, tükrözéssel, zoomolással, eltolással, fényerő szabályozásával
- Így valójában a modell általánosítóképességét javítjuk



Keras ImageDataGenerator

- Keras esetén elérhető az `ImageDataGenerator` osztály, amely a képfeldolgozásban is segít

```
from keras.preprocessing.image import ImageDataGenerator

train_gen = ImageDataGenerator(
        rescale=1./255,
        featurewise_std_normalization=True,
        rotation_range=30,
        width_shift_range=0.1,
        height_shift_range=0.1,
        zoom_range=0.1,
        vertical_flip=False,
        horizontal_flip=True,
        brightness_range=[0.75, 1.25]
)
```

Képek betöltése

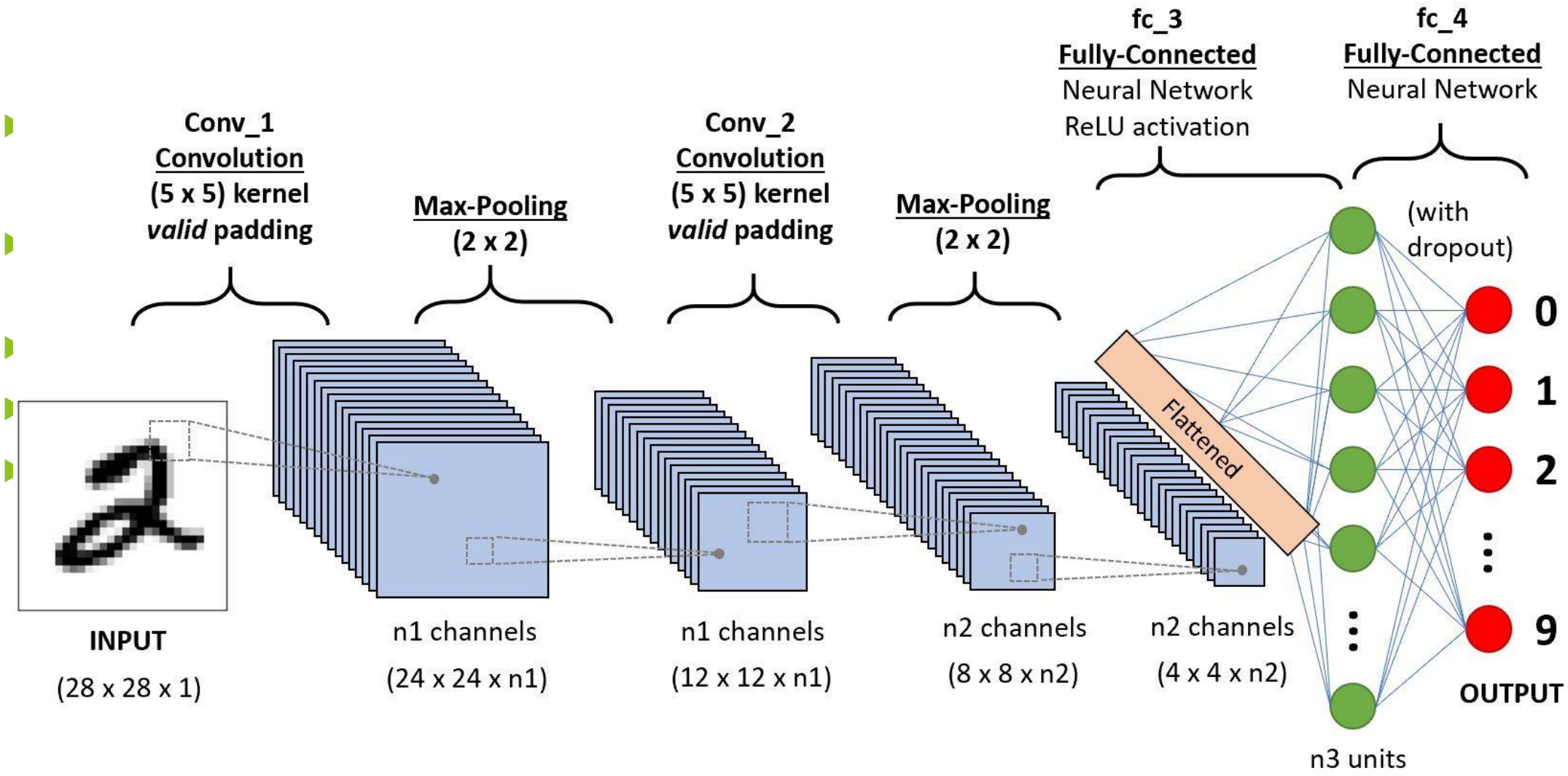
- A generátornak a `.flow_from_directory()` metódusa segítségével egy adott könyvtárban található képek végigiterálhatóak, a hálózat bemeneteként megadhatók
- A Keras Modelnek pedig van `.fit_generator()` metódusa, amellyel összeköthető tanításkor

```
model.fit_generator(
        train_gen.flow_from_directory(
                "./train",
                target_size=(imsize, imsize),
                batch_size=batch_size
        ),
        epochs=20
)
```

Lépések

1. Képek kezelése
   - Tanítás és validálás
2. Hálózati architektúra kidolgozása
   - Többosztályú klasszifikáció
3. Tanítás
   - Nyomonkövetés

A hálózat architektúrája



Többosztályú klasszifikáció

- A kimenet nem lehet egyetlen numerikus érték
- A kategóriák között nincs folytonossági kapcsolat
  - Pl. Kutya-Macska közötti döntés: van átmenet? Mi történik ha egér a bemenet? Az „köztes” állapot?

0

1

?

0.5

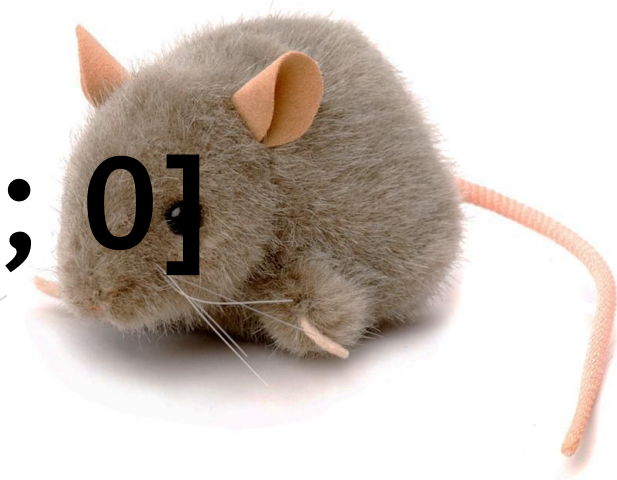
Többosztályú klasszifikáció

- A kimenet legyen egy vektor
- Az egyes dimenziók más kategóriákat reprezentálnak
- Egy magas érték magas valószínűséget prediktál

[1; 0]

[0; 1]

[0; 0]



Prediktáláskor

**„Inkább macska, mint kutya
[0.88, 0.12]**

Konvolúciós architektúra Keras-ban

```
model.summary()
```

```
Layer (type)                 Output Shape              Param #
conv2d_1 (Conv2D)            (None, 122, 122, 32)      4736
max_pooling2d_1 (MaxPooling2 (None, 61, 61, 32)        0
conv2d_2 (Conv2D)            (None, 57, 57, 64)        51264
max_pooling2d_2 (MaxPooling2 (None, 28, 28, 64)        0
conv2d_3 (Conv2D)            (None, 26, 26, 128)       73856
max_pooling2d_3 (MaxPooling2 (None, 13, 13, 128)       0
conv2d_4 (Conv2D)            (None, 11, 11, 256)       295168
max_pooling2d_4 (MaxPooling2 (None, 5, 5, 256)         0
flatten_1 (Flatten)          (None, 6400)              0
dense_1 (Dense)              (None, 256)               1638656
dense_2 (Dense)              (None, 256)               65792
dense_3 (Dense)              (None, 64)                16448
dense_4 (Dense)              (None, 5)                 325
Total params: 2,146,245
Trainable params: 2,146,245
Non-trainable params: 0
```

Lépések

1. Képek kezelése
   - Tanítás és validálás
2. Hálózati architektúra kidolgozása
   - Többosztályú klasszifikáció
3. Tanítás
   - Nyomonkövetés

Tanítás

```
model.compile(optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy'])

from tensorflow.keras.callbacks import TensorBoard

tbcb = TensorBoard("logs/sample",
histogram_freq=0)
model.fit_generator(train_data,
            epochs=200,
            validation_data=test_data,
            callbacks=[tbcb]
)
```

Teljesítmény

- Validációra használjuk a kézzel kiválogatott 250 képet, a maradék 750el tanítunk
  - A 750 képet megmutatjuk a modellnek, és hibavisszaterjesztéssel módosítjuk a súlyokat
  - Egy ilyen kört „epoch”nak nevezünk
  - Minden epoch végén validálunk, megnézzük a pontosságot azokra a képekre, amelyekkel nem végzünk tanítást

Pontosság

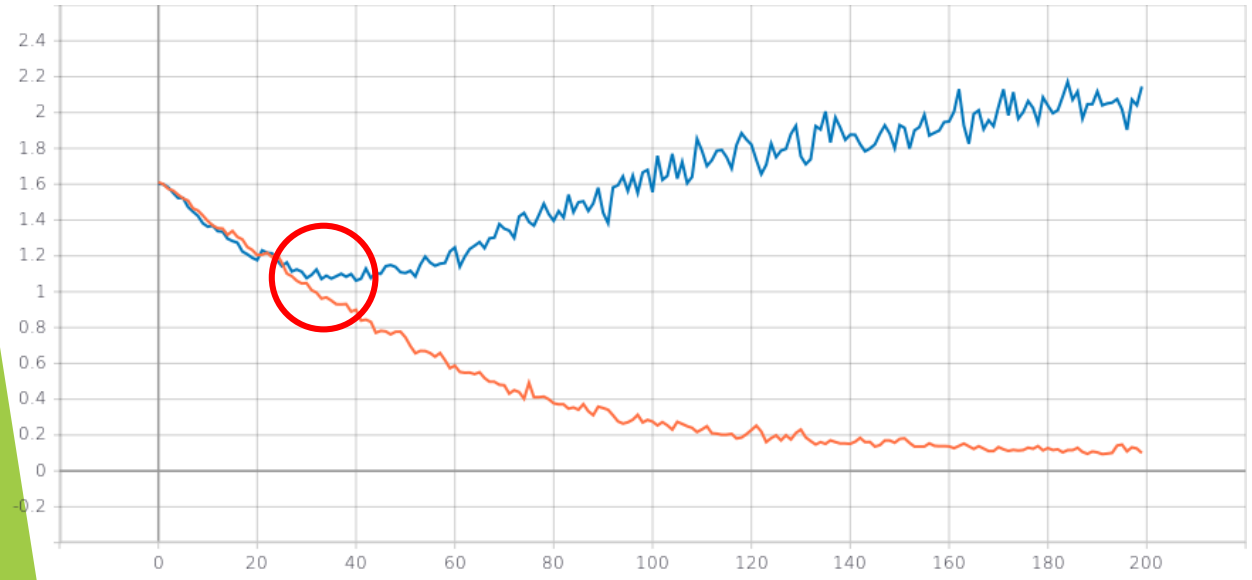
- 200 epoch után azt látjuk, hogy:

```
loss: 0.1002 - accuracy: 0.9685 - val_loss: 2.145 -
val_accuracy: 0.6064
```

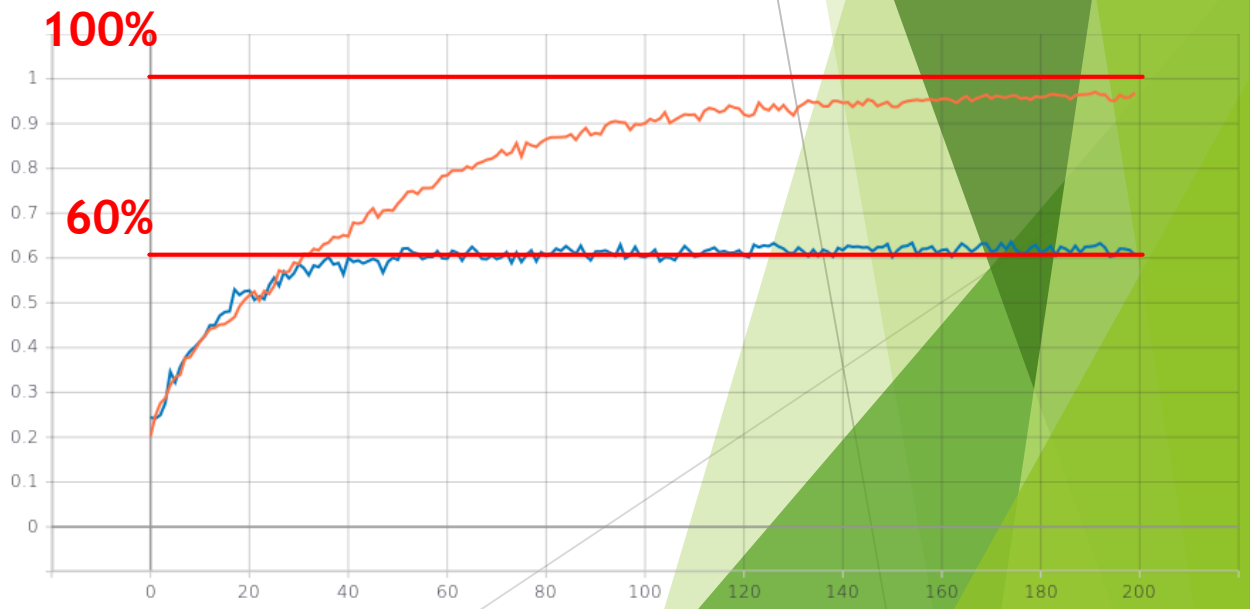
- Azaz a tanítómintákra mért pontosság 96,85%
- A validációs adatokra csak 60,64%
  - Ez jelentős különbség!
- A veszteségfüggvény esetén ugyanez a kettősség látszik

Teljesítmény

- Vizsgáljuk meg ezen értékek alakulását az időben

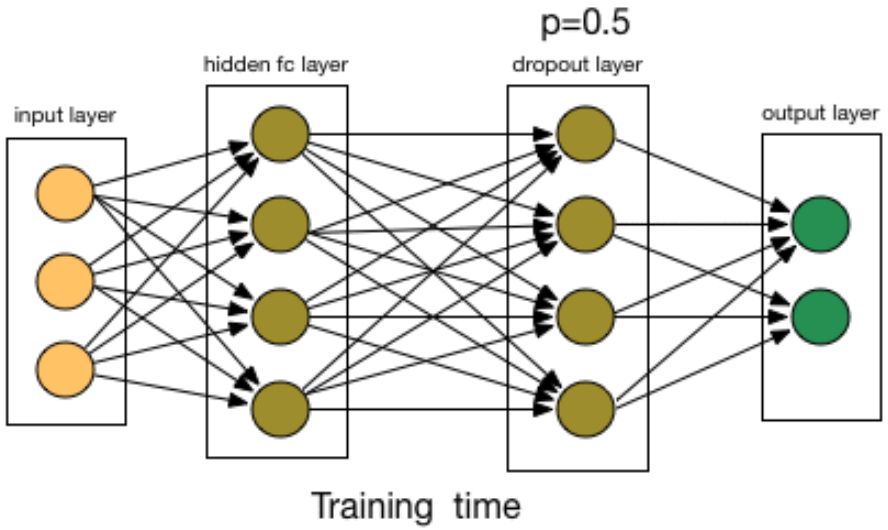


Loss



Accuracy

Overfitting



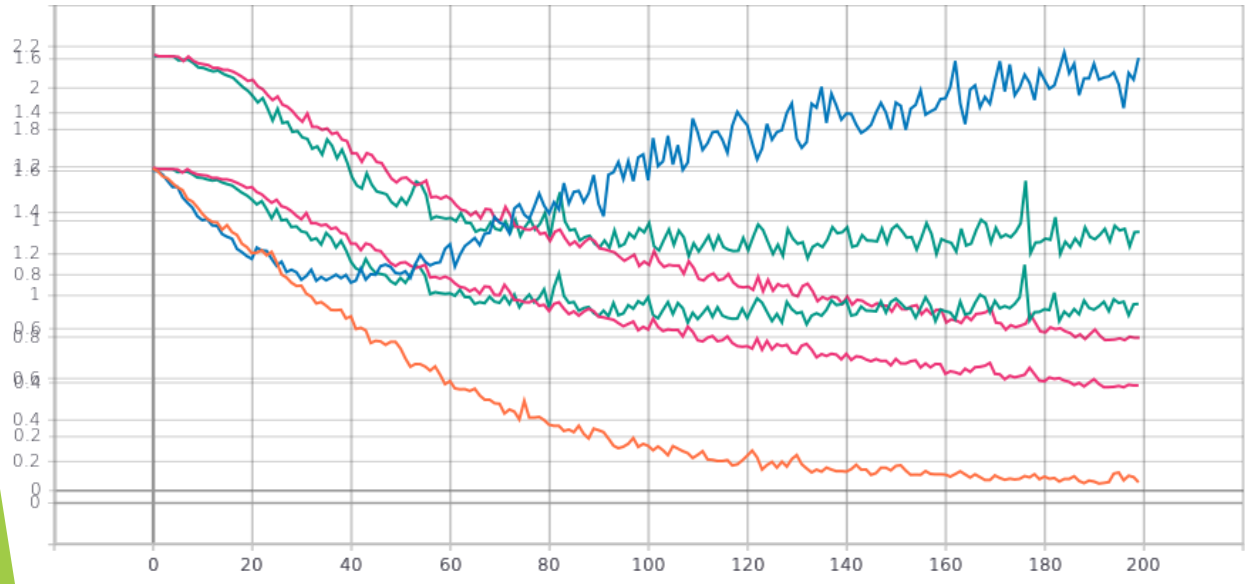
- Dropout
  - A neuronok adott valószínűség szerint kiesnek, nem vesznek részt a tanulásban
  - Így a fix aktivációk helyett minden neuronnak ismerni kell kissé a teljes probléma egy részét

Ugyanez a modell Dropout-tal Kerasban

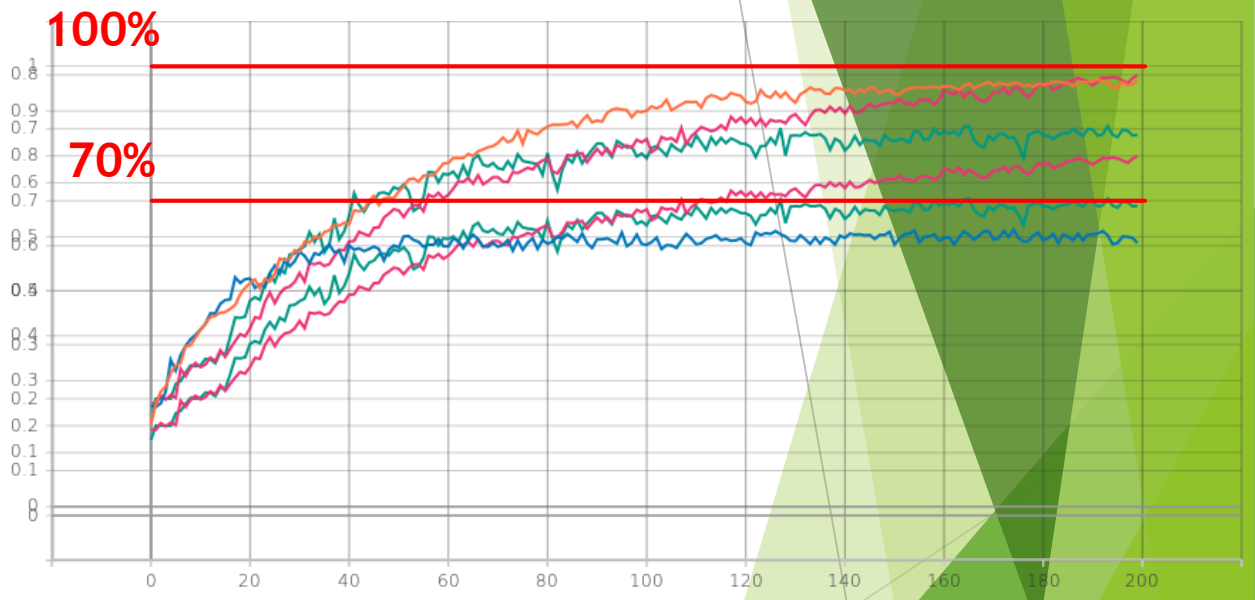
```
model = Sequential()
model.add(Convolution2D(32, (7, 7),
        input_shape=(imsize,imsize,3),
        activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Convolution2D(64, (5,
5),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Convolution2D(128, (3,
3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

```
model.add(Convolution2D(256, (3, 3),
            activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(5, activation='softmax'))
```

Teljesítmény



Loss



Accuracy

Eredmények értékelése

- Alapesetben 20% a random klasszifikálás
- A túlillesztett modell 20 perc alatt 60% körüli pontosságot ér el, majd stagnál (a legjobb érték 63% körüli)
- Dropout alkalmazásakor más a hatás, a legjobb elért pontosság kicsivel jobb, 68% körüli
- Élesben ez persze még nagyon kevés: a cél a 90% feletti pontosság
  - Az elméleti határ az emberi hiba alatti

Növeljük a teljesítményt

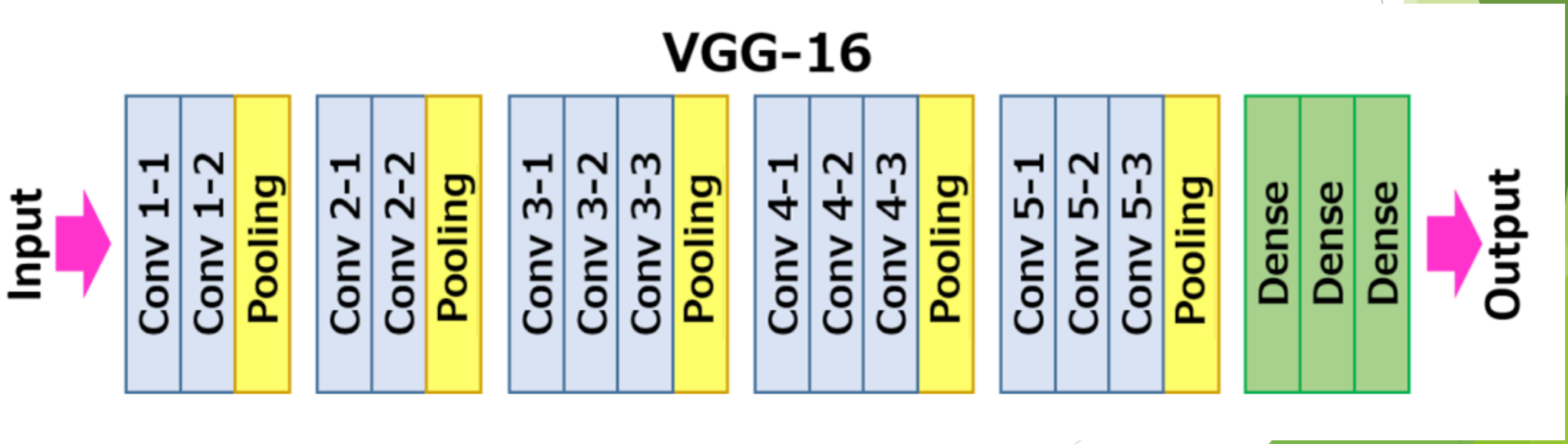
- Nagyobb bemeneti képméret
- Nagyobb epoch-szám
- Több tanítható paraméter a hálózatban, nagyobb „mélység”
- Sokkal több tanítóminta

- Ezek mind lassítják a tanítás folyamatát

ELKH Cloud

Bevált architektúrák

- VGG-16 az ImageNet challengen jó eredményeket ért el
- Használjuk ezt a jól bevált architektúrát!



VGG-16 architektúra

```
from tensorflow.keras.applications.vgg16
import VGG16
model = VGG16(include_top=True, weights=None,
classes=5)
model.compile(optimizer='sgd',
      loss='categorical_crossentropy',
      metrics=['accuracy'])
```

VGG-16 architektúra

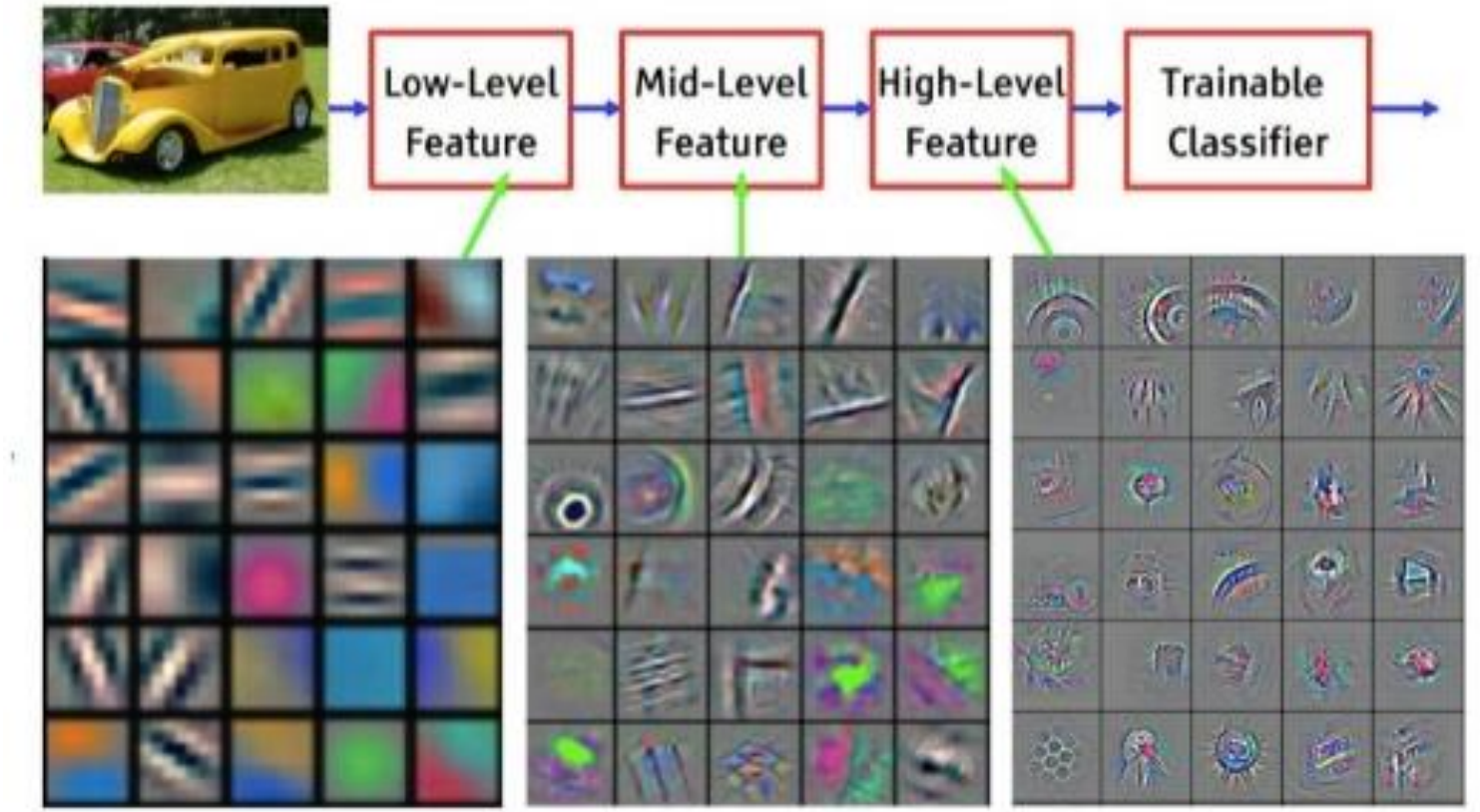
Loss

70%

Accuracy

Mit tanulnak a konvolúciós rétegek?

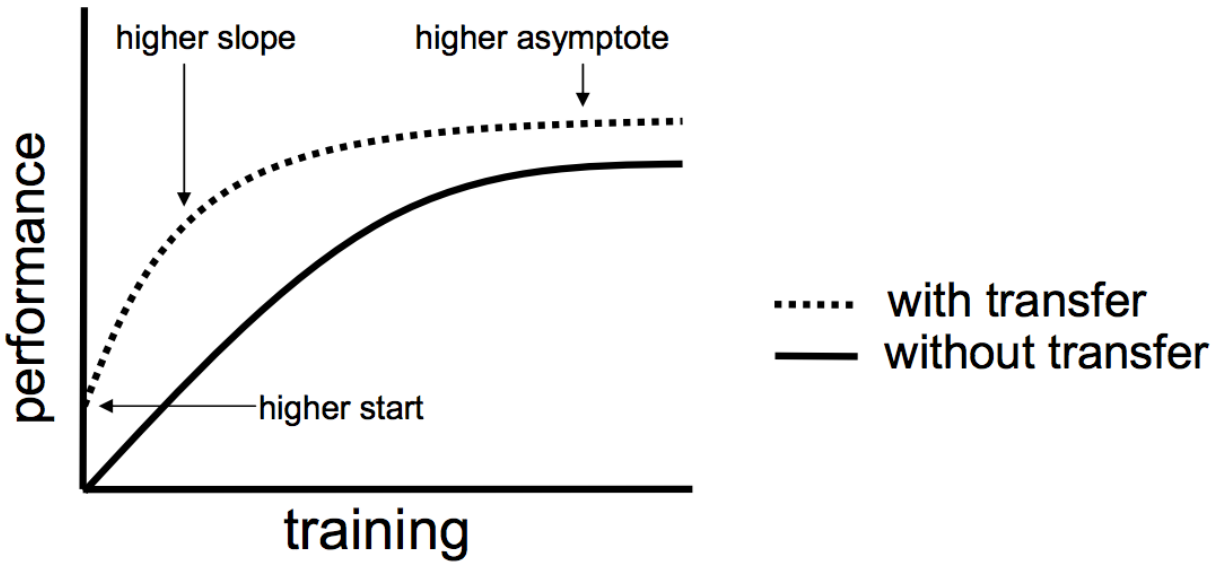
- Mélyebb kerneleket vizsgálva egyre összetettebb jellemzők



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Transfer learning

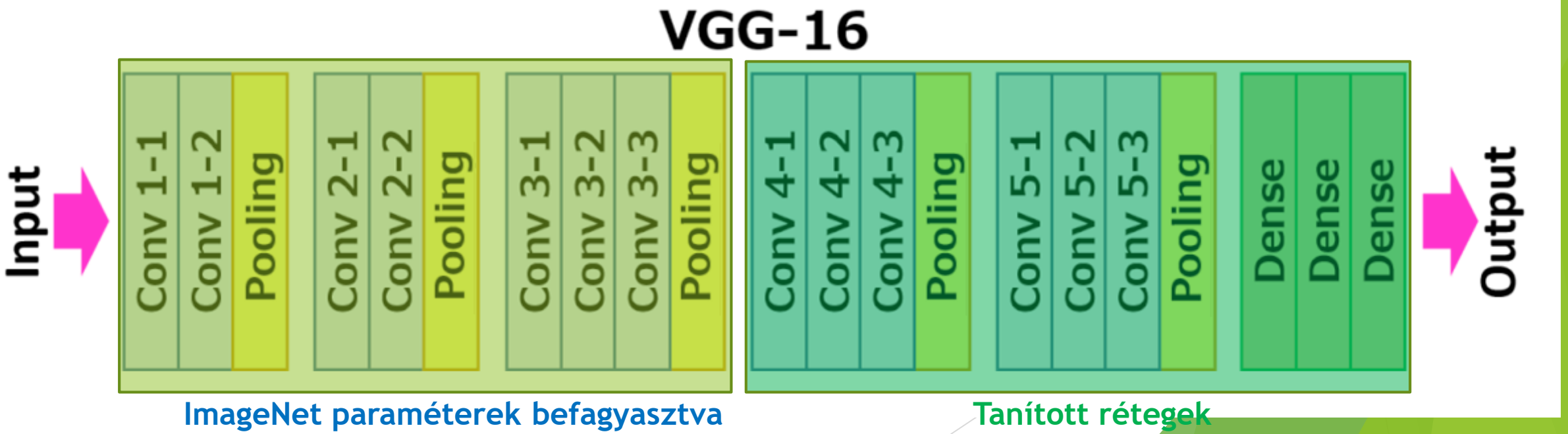
- Célszerű lehet a magasabban levő jellemzőket felismerő rétegeket létező kész implementációkból felhasználni
- Például a korábban említett ImageNet challengere betanított hálózat értékei szerint



https://machinelearningmastery.com/transfer-learning-for-deep-lear

Transfer learning

- VGG-16 alapján
- A modell felső rétegeit nem tanítjuk, csak a későbbi rétegek paramétereit
  - Vagy akár új rétegeket is köthetünk a hátsó részekre



Transfer learning Kerasban

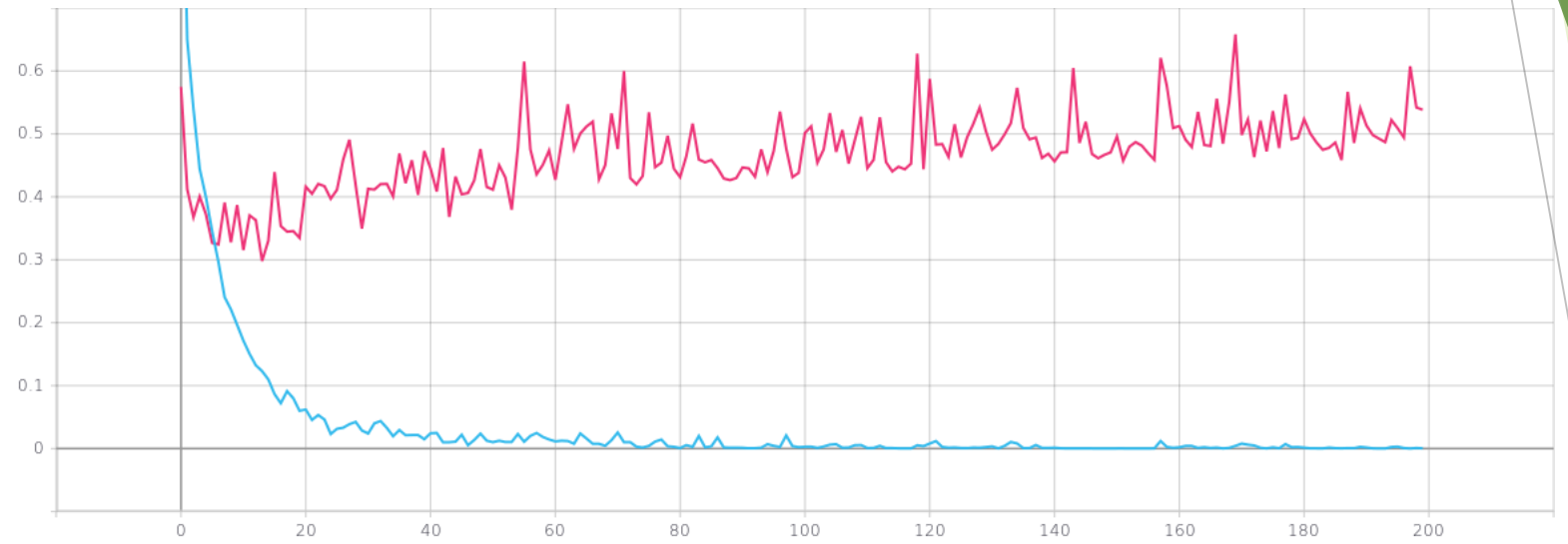
```
from tensorflow.keras.applications.vgg16 import VGG16
vgg16 = VGG16(include_top=True, weights='imagenet', classes=1000)

for layer in vgg16.layers[:12]:
    layer.trainable=False
for layer in vgg16.layers[12:]:
    layer.trainable=True
```

Transfer learning Kerasban

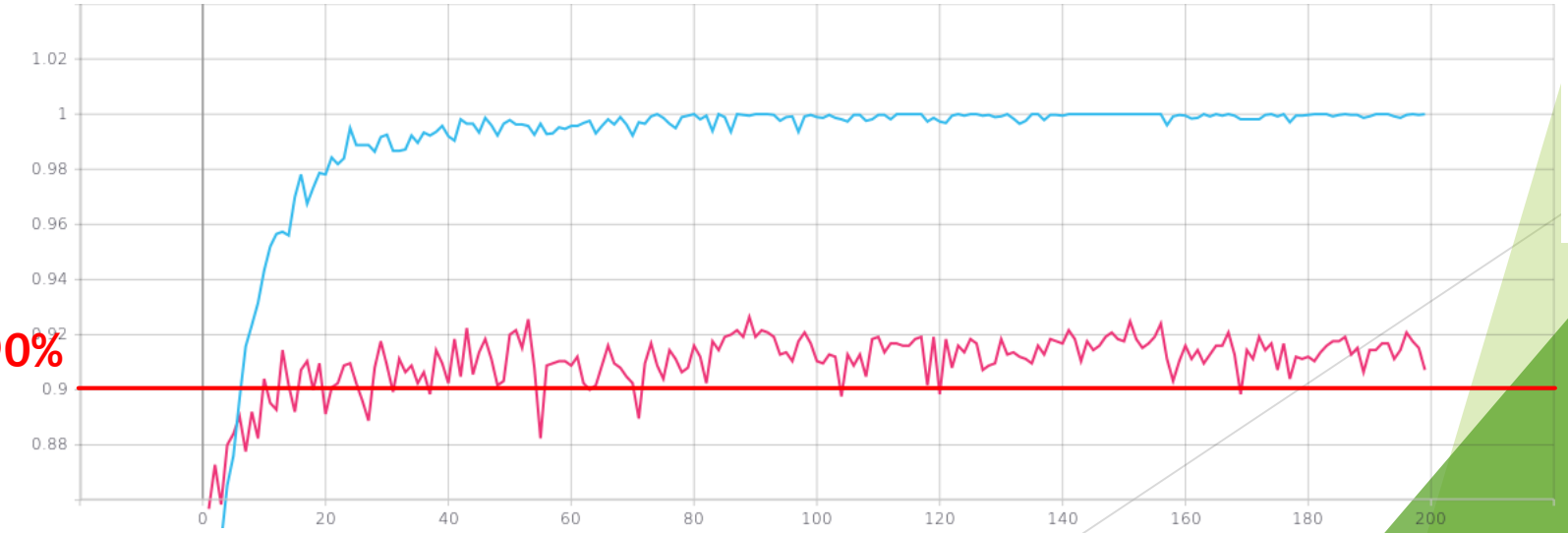
```
fc1 = vgg16.layers[-3]
fc2 = vgg16.layers[-2]
predictions = vgg16.layers[-1]
dropout1 = Dropout(0.25)
dropout2 = Dropout(0.25)

x = dropout1(fc1.output)
x = fc2(x)
x = dropout2(x)
predictors = predictions(x)

model_ = Model(inputs=vgg16.input, outputs=predictors)
x = Dense(5, activation='softmax', name='predictions')(model_.layers[-
2].output)
model = Model(inputs=model_.input, outputs=x)
```

Transfer learning eredmények

Loss



Accuracy



90%

Dropout regularizáció

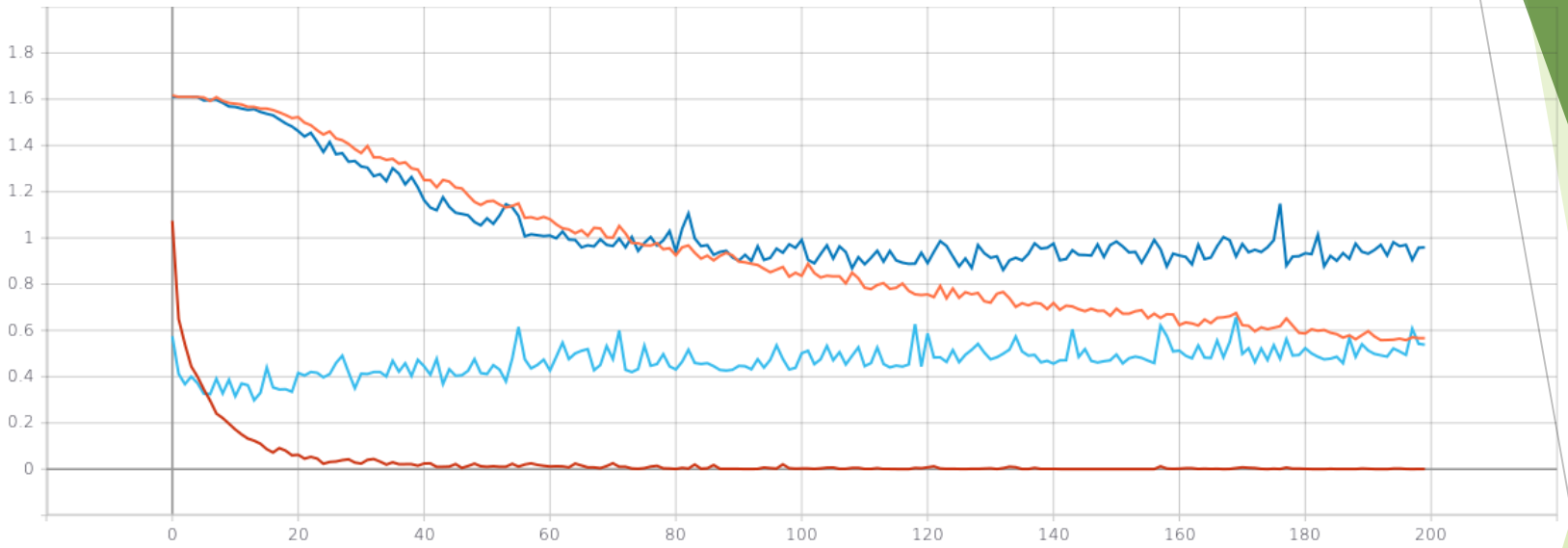
Validációs pontosság:

92%

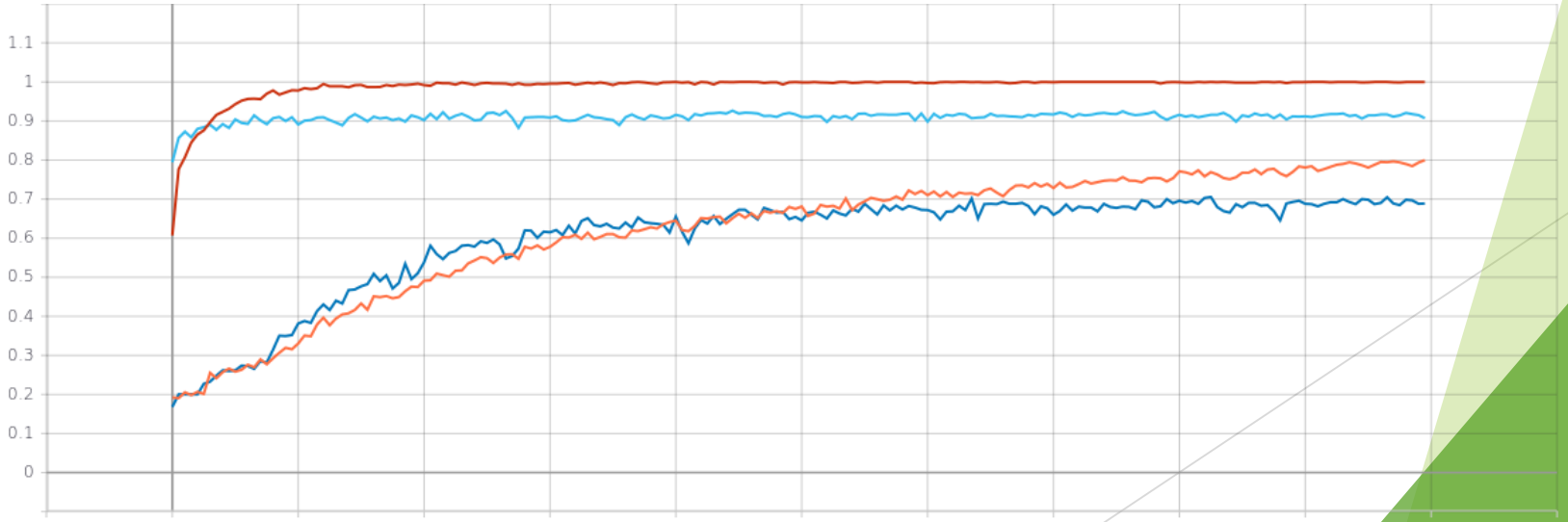
Összehasonlításképp

ELKH Cloud

Loss



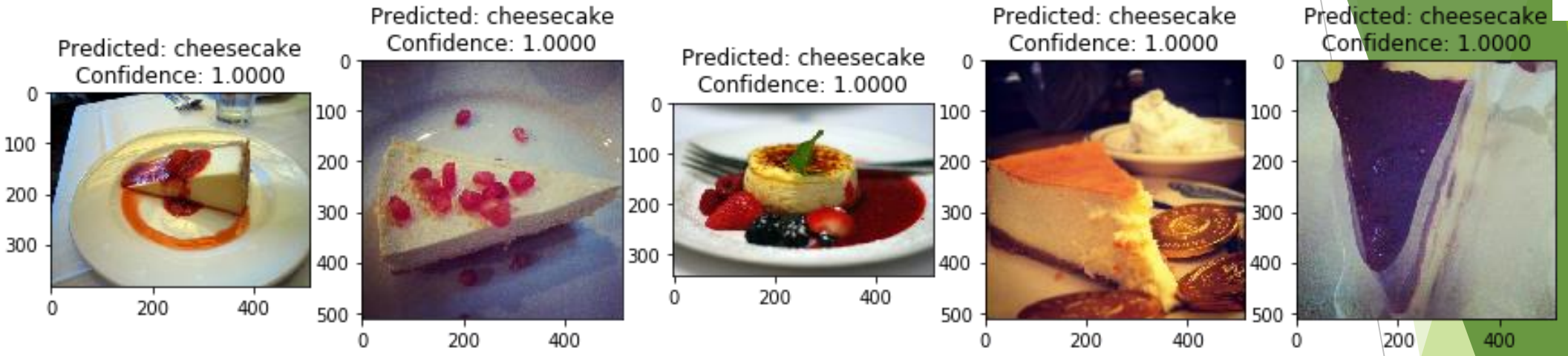
Accuracy

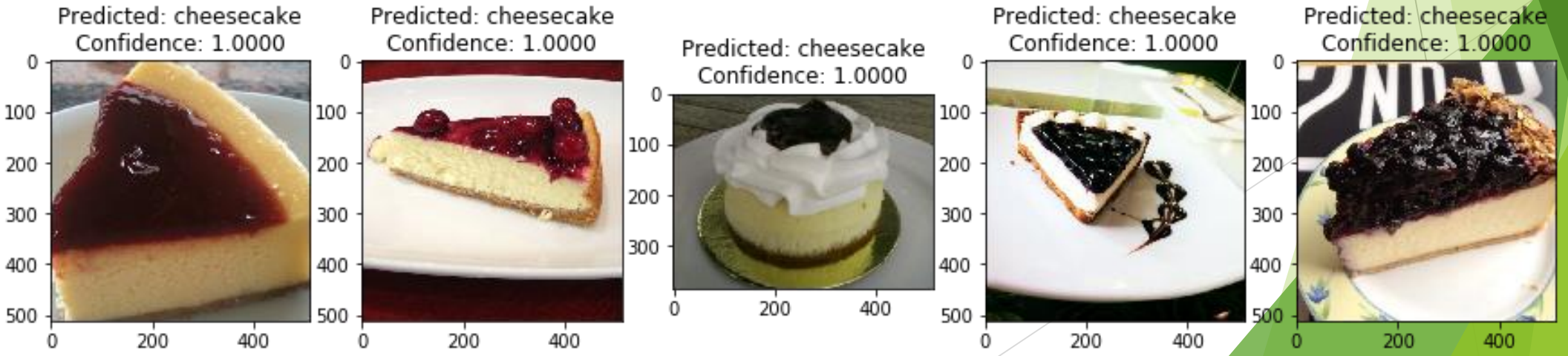


Eredmények értékelése

- Helyes klasszifikáció, magas magabiztossággal
- Helyes klasszifikáció, alacsony magabiztossággal
- Helytelen klasszifikáció, alacsony magabiztossággal
- Helytelen klasszifikáció, magas magabiztossággal

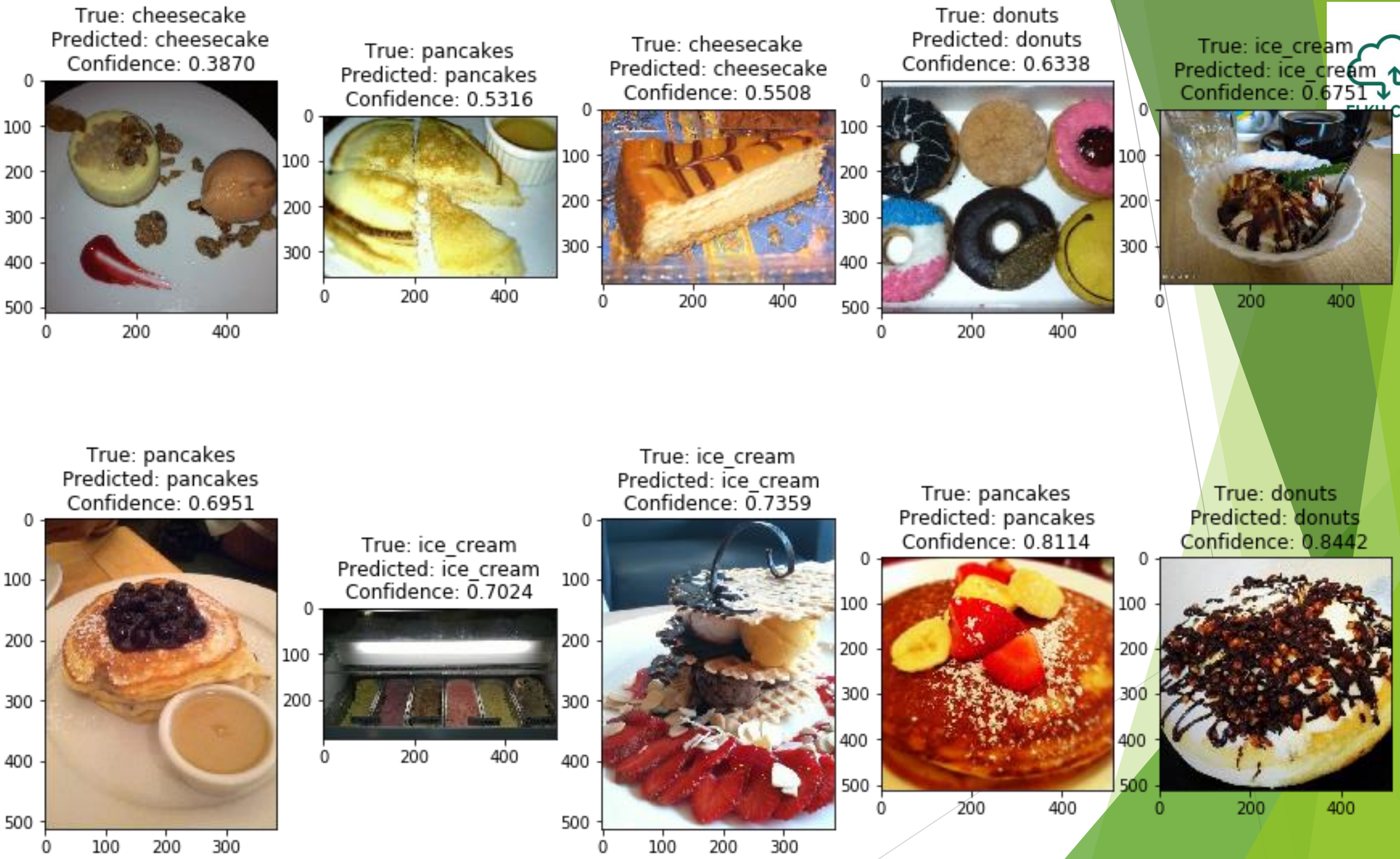
cheesecake


Predicted: cheesecake
Confidence: 1.0000
Predicted: cheesecake
Confidence: 1.0000
Predicted: cheesecake
Confidence: 1.0000
Predicted: cheesecake
Confidence: 1.0000
Predicted: cheesecake
Confidence: 1.0000


Predicted: cheesecake
Confidence: 1.0000
Predicted: cheesecake
Confidence: 1.0000
Predicted: cheesecake
Confidence: 1.0000
Predicted: cheesecake
Confidence: 1.0000
Predicted: cheesecake
Confidence: 1.0000

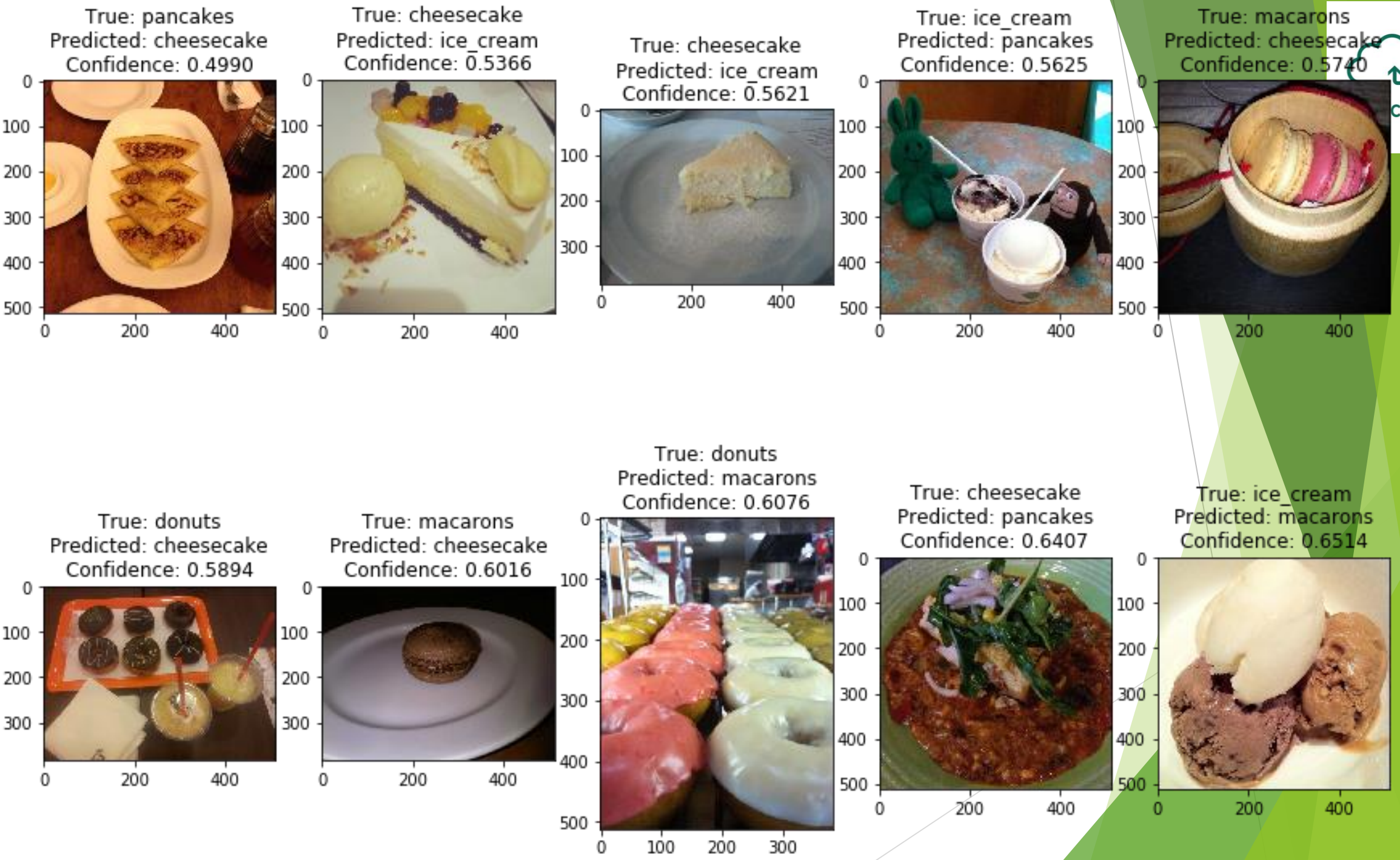
Eredmények értékelése

- Helyes klasszifikáció, magas magabiztossággal
- Helyes klasszifikáció, alacsony magabiztossággal
- Helytelen klasszifikáció, alacsony magabiztossággal
- Helytelen klasszifikáció, magas magabiztossággal


True: cheesecake
Predicted: cheesecake
Confidence: 0.3870
True: pancakes
Predicted: pancakes
Confidence: 0.5316
True: cheesecake
Predicted: cheesecake
Confidence: 0.5508
True: donuts
Predicted: donuts
Confidence: 0.6338
True: ice_cream
Predicted: ice_cream
Confidence: 0.6751
True: pancakes
Predicted: pancakes
Confidence: 0.6951
True: ice_cream
Predicted: ice_cream
Confidence: 0.7024
True: ice_cream
Predicted: ice_cream
Confidence: 0.7359
True: pancakes
Predicted: pancakes
Confidence: 0.8114
True: donuts
Predicted: donuts
Confidence: 0.8442

Eredmények értékelése

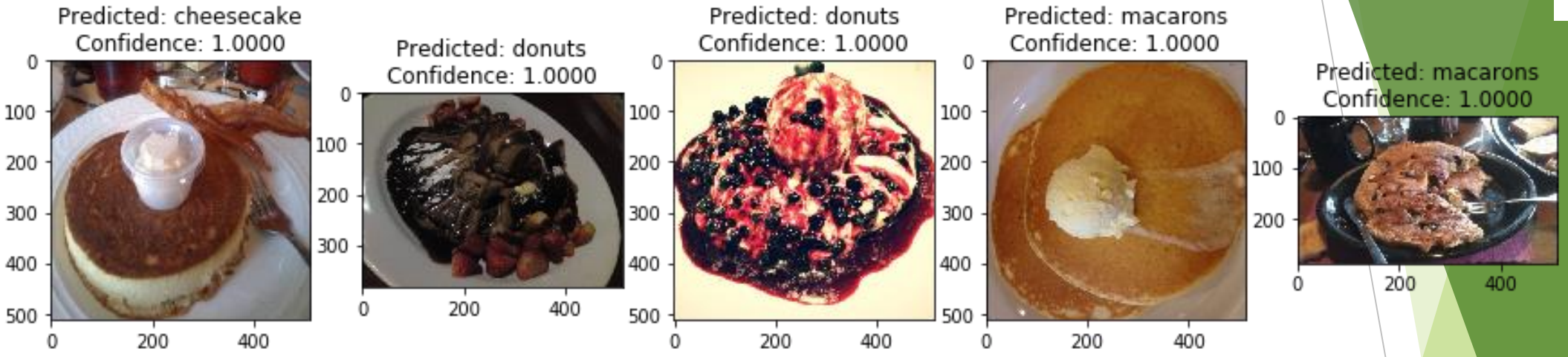
- Helyes klasszifikáció, magas magabiztossággal
- Helyes klasszifikáció, alacsony magabiztossággal
- Helytelen klasszifikáció, alacsony magabiztossággal
- Helytelen klasszifikáció, magas magabiztossággal

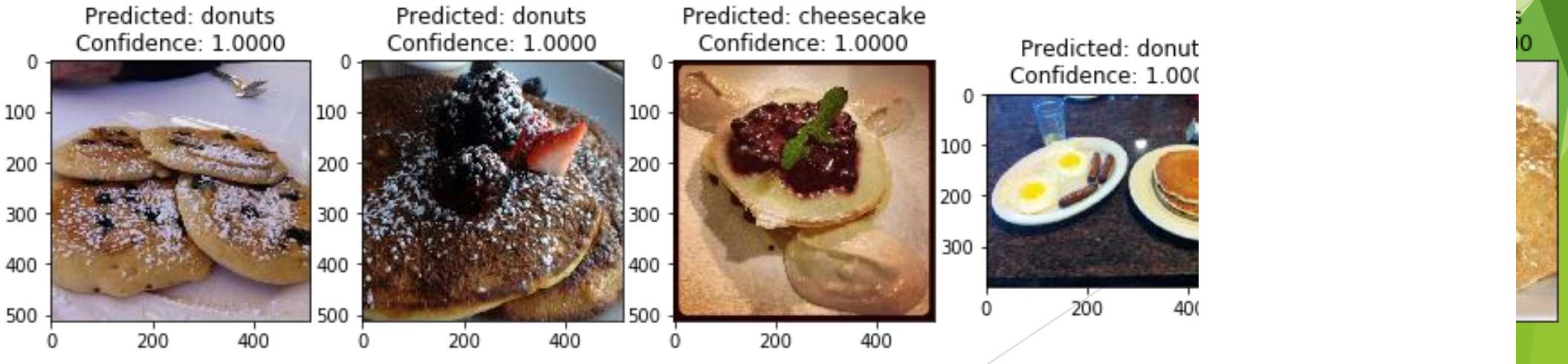

True: pancakes
Predicted: cheesecake
Confidence: 0.4990
True: cheesecake
Predicted: ice_cream
Confidence: 0.5366
True: cheesecake
Predicted: ice_cream
Confidence: 0.5621
True: ice_cream
Predicted: pancakes
Confidence: 0.5625
True: macarons
Predicted: cheesecake
Confidence: 0.5740
True: donuts
Predicted: cheesecake
Confidence: 0.5894
True: macarons
Predicted: cheesecake
Confidence: 0.6016
True: donuts
Predicted: macarons
Confidence: 0.6076
True: cheesecake
Predicted: pancakes
Confidence: 0.6407
True: ice_cream
Predicted: macarons
Confidence: 0.6514

Eredmények értékelése

- Helyes klasszifikáció, magas magabiztossággal
- Helyes klasszifikáció, alacsony magabiztossággal
- Helytelen klasszifikáció, alacsony magabiztossággal
- Helytelen klasszifikáció, magas magabiztossággal

pancakes


Predicted: cheesecake
Confidence: 1.0000
Predicted: donuts
Confidence: 1.0000
Predicted: donuts
Confidence: 1.0000
Predicted: macarons
Confidence: 1.0000
Predicted: macarons
Confidence: 1.0000


Predicted: donuts
Confidence: 1.0000
Predicted: donuts
Confidence: 1.0000
Predicted: cheesecake
Confidence: 1.0000
Predicted: donut
Confidence: 1.000

Futásidők

- Az eredeti „naiv” modell esetén
  - 1 óra 37 perc volt a 200 epoch futtatása
- A VGG-16 modell túlilleszkedett ~80 epoch után
  - Így is több mint 5 óra volt!
- Ugyanezen modell transfer learning felépítés esetén kicsivel gyorsabb
  - De 200 epochon át tanítani több mint 9 órán át tart

- És ez 3750 kép mint tanítóminta, nem BIG DATA!

GPU feldolgozás

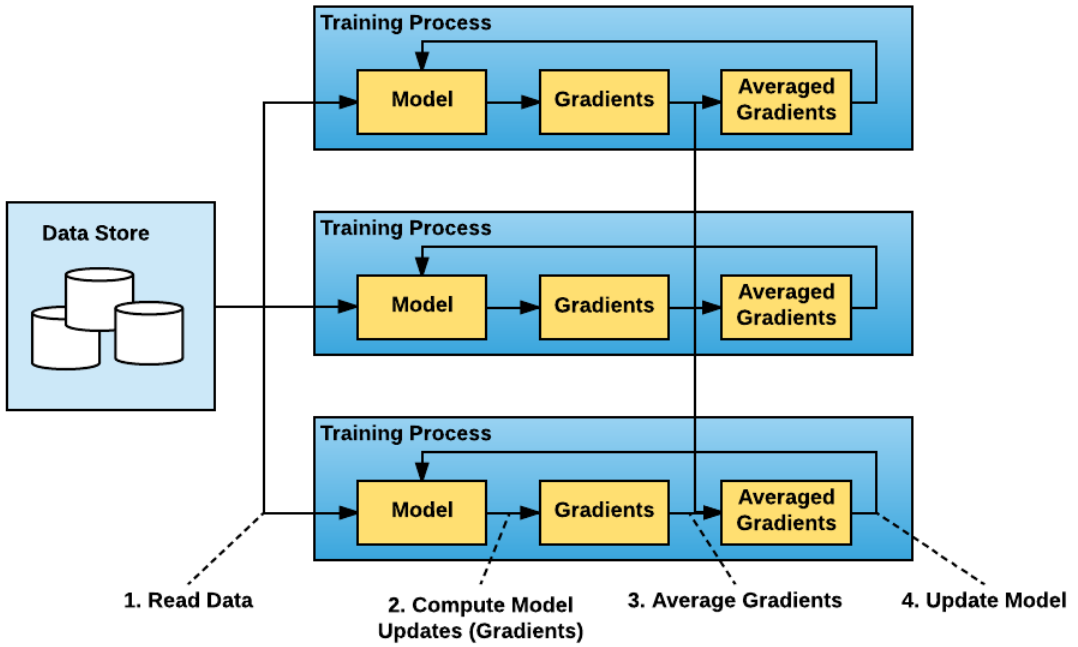
- A TensorFlow alapesetben GPU-t használ, ha tud
  - `tensorflow-gpu` package kell hozzá és CUDA kompatibilis grafikus gyorsító
- A modell paraméterei és a tanítóminták az eszközre másolódnak, majd a mátrixműveletek párhuzamosan kerülnek végrehajtásra
- A modelleket GPUn tanítottam (Tesla K20Xm)
  - Teszteltem, CPU használatával nagyjából négyszer tovább tartott volna

Több GPU?

- Több GPU használatával a feldolgozás ideje elvben csökkenthető
  - Adatok szerinti felbontással az egyes gyorsítók a különböző részhalmazzal tanítják a modelljük másolatát
  - Azonban ekkor a GPU-k közötti kommunikáció lesz a szűk keresztmetszet, a paraméterek frissítésekor
- A Keras beépítve támogatja az adatpárhuzamos felbontást: `multi_gpu_model()`
- Teszteltem 4 GPUs környezetben
  - Valamivel kevesebb mint 4 óra volt a VGG-16 transfer learninges megoldás feldolgozása
  - Tehát 2.5× gyorsítás

Még több GPU??

- Ha az adatok mennyisége indokolja, akár GPU-klaszterben is kivitelezhető a megoldás, elosztott formában
- Például a Horovod library alkalmazható e célra
  - A bemutatott kód minimálisan kiegészítendő
  - Ezt követően MPI protokoll szerint kommunikálnak a nodeok



https://eng.uber.com/horovod/

Köszönöm a figyelmet!

kertesz.gabor@sztaki.hu