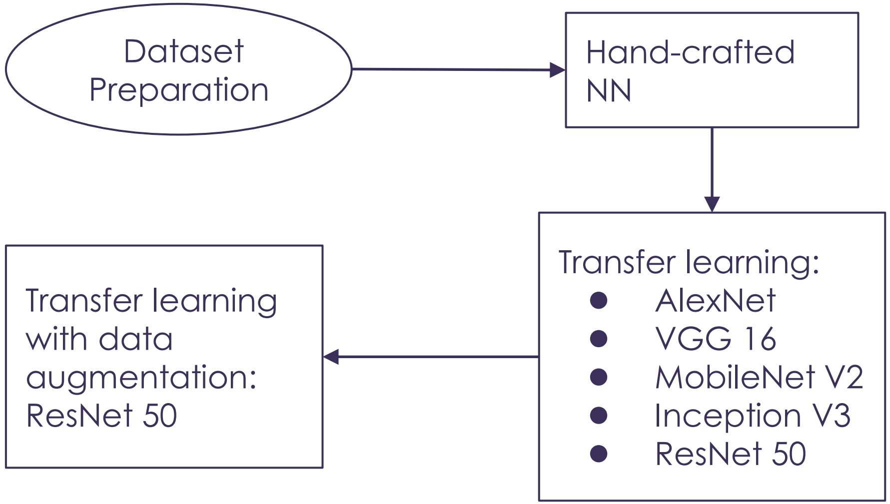
Foundations of Deep Learning
Image classification

A.A. 2021/2022

EPIFANI ARMANDO 826153
GIORGETTI GLORIA 826226
STOFFA GIACOMO 830159

Index

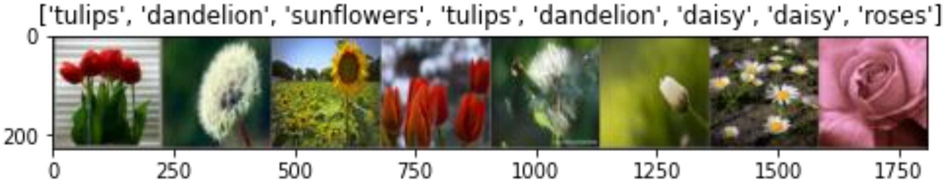
- Dataset
- Hand-crafted NN
- Transfer Learning
- Compare Models
- Data Augmentation
- Results
- Conclusion



Dataset

Task: image classification on a flower dataset composed of 3670 divided in 5 classes:

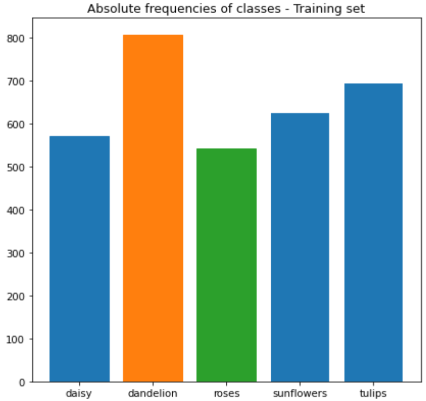
- daisy (633)
- dandelion(898)
- roses (641)
- sunflowers (699)
- tulips(799)

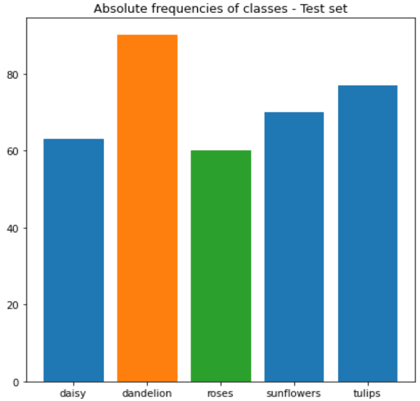


Dataset

Removed Extra images (73) (supervised), in particular from classes "roses" and "tulips"

- TRAIN (~70%) → 2525
- VALIDATION (~20%) → 712
- TEST (10%) → 360

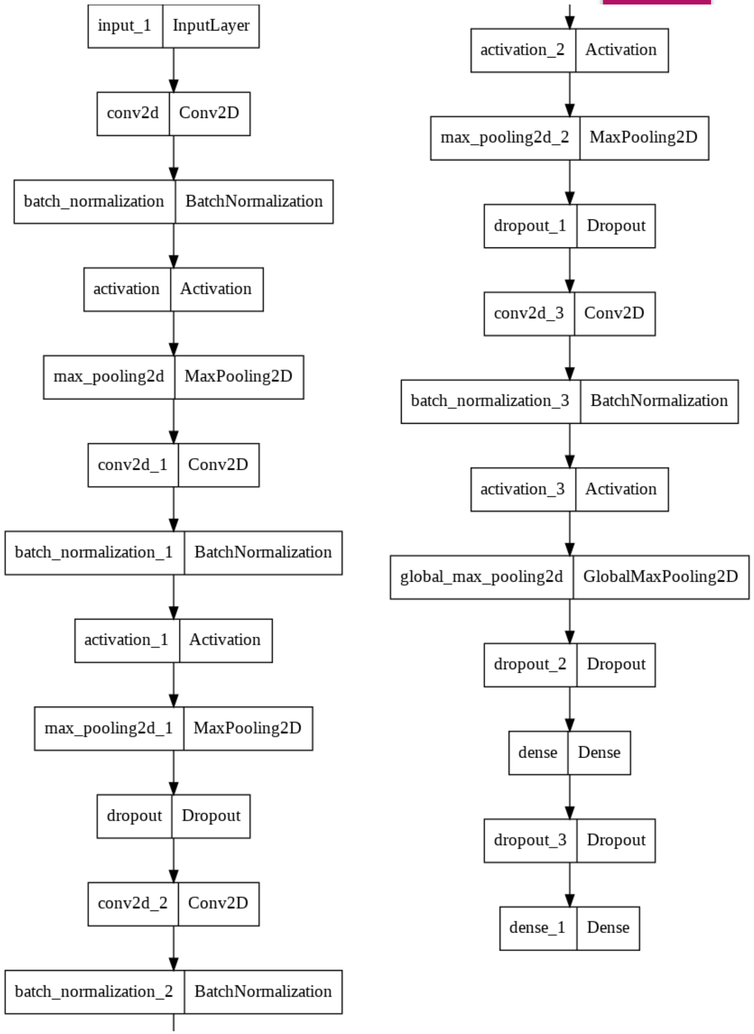




Architecture

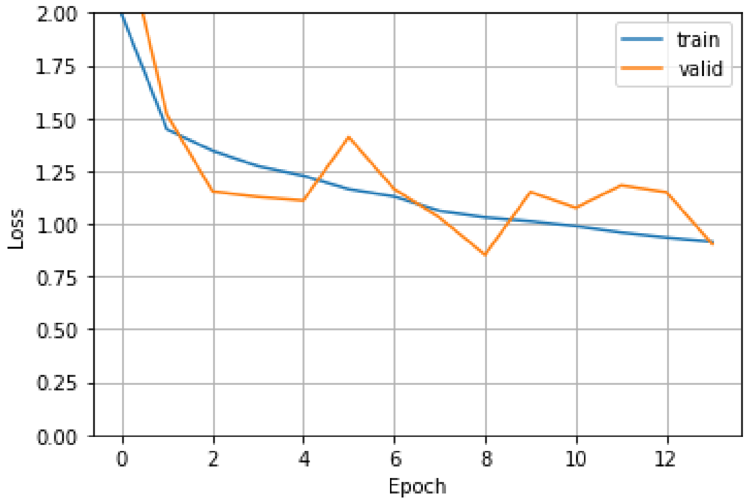
- No data augmentation
- Train/validation batch size=32
- Respectively 32, 64, 128, 256 neurons for each group (L2-type regularization)
- 64 neurons dense layer before last dropout
- Dropout
- Dense layer with softmax activation as last layer mapped on number of classes

- Total params: 407,109
- Trainable params: 406,149
- Non-trainable params: 960



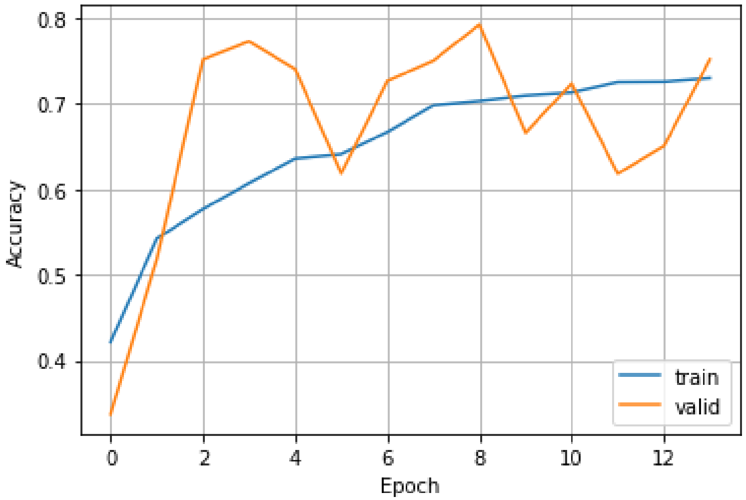
Training

- 30 epochs, categorical cross-entropy as loss function, Adam as optimizer
- Other metrics: accuracy
- Early stopping (patience on validation loss equal to 5) made training stop after 14 epochs
- Training set
  - loss 0.91
  - accuracy 72.95%
- Validation set
  - loss 0.90
  - accuracy 75.14%



Training

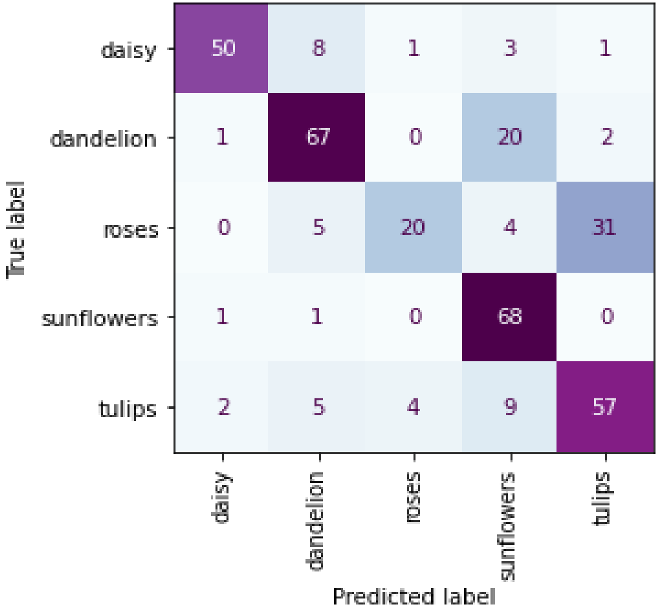
- 30 epochs, categorical cross-entropy as loss function, Adam as optimizer
- Other metrics: accuracy
- Early stopping (patience on validation loss equal to 5) made training stop after 14 epochs
- Training set
  - loss 0.91
  - accuracy 72.95%
- Validation set
  - loss 0.90
  - accuracy 75.14%



Results on test set

- 98 images out of 360 misclassified.
- Test set
  - loss 0.88
  - accuracy 72.78%

	precision	recall	f1-score	support
daisy	0.93	0.79	0.85	63
dandelion	0.78	0.74	0.76	90
roses	0.80	0.33	0.47	60
sunflowers	0.65	0.97	0.78	70
tulips	0.63	0.74	0.68	77



Results on test set

True class	Predicted class	% daisy	% dandelion	% roses	% sunflowers	% tulips
daisy	daisy	43.78	27.50	5.65	21.65	1.42

True class	Predicted class	% daisy	% dandelion	% roses	% sunflowers	% tulips
roses	tulips	0.77	0.55	42.55	0.86	55.27

Results on test set

True class	Predicted class	% daisy	% dandelion	% roses	% sunflowers	% tulips
daisy	dandelion	13.50	81.69	3.10	0.62	1.09

True class	Predicted class	% daisy	% dandelion	% roses	% sunflowers	% tulips
dandelion	dandelion	0.08	58.84	0.01	40.35	0.70

Transfer Learning

- Transfer Learning: improves performance by transferring the knowledge acquired from a task to a different task.
- Dataset of 2500 images: convolutional blocks' weights of the pre-trained network remain the same, while fully connected layers are modified and re-trained.
- Imagenet: dataset of almost one million images and one thousand classes (like daisy, rapeseed and the orchid Cymbidium parviflorum).
- Networks used for transfer learning were developed specifically for classification of Imagenet dataset during ILVRC.

Comparison

	Training set		Validation set		Test set	
	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
Hand-crafted	0.91	72.95	0.90	75.14	0.88	72.78
MobileNet V2	0.02	99.79	0.40	89.47	0.32	91.11
AlexNet	0.09	96.57	0.06	97.71	0.35	89.72
Inception V3	0.34	87.80	0.40	87.08	0.41	87.22
VGG 16	0.28	89.86	0.39	89.47	0.31	89.72
ResNet 50	0.20	93.15	0.31	90.59	0.20	93.06

Architecture

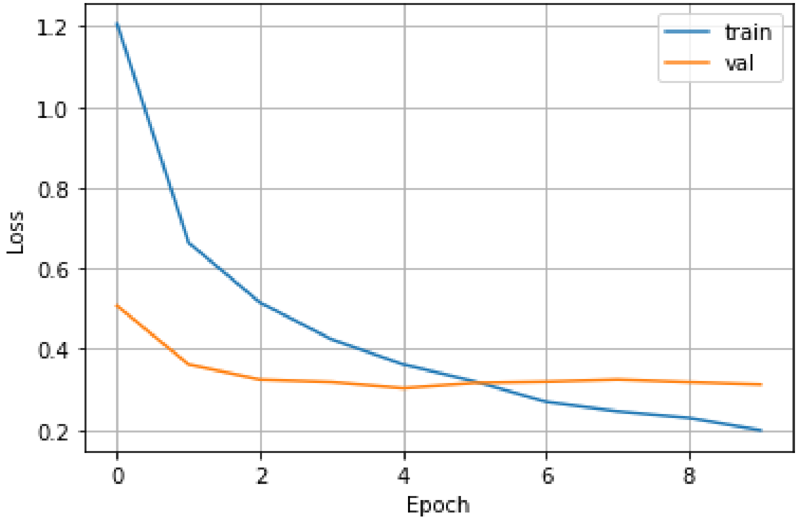
- ResNet-50 shows the best performances.
- ResNet-50: residual blocks and dense output layer (softmax activation).
- Output layer of ResNet-50 removed, fully connected layers and dropout layers added.

- Trainable parameters: ~ 210 thousand
- Non-trainable parameters: ~24 millions

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 224, 224, 3)]     0

 tf.__operators__.getitem (S  (None, 224, 224, 3)      0
 licingOpLambda)

 tf.nn.bias_add (TFOpLambda)  (None, 224, 224, 3)      0

 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 100)               204900

 dropout (Dropout)           (None, 100)               0

 dense_1 (Dense)             (None, 50)                5050

 dropout_1 (Dropout)         (None, 50)                0

 dense_2 (Dense)             (None, 5)                 255
```

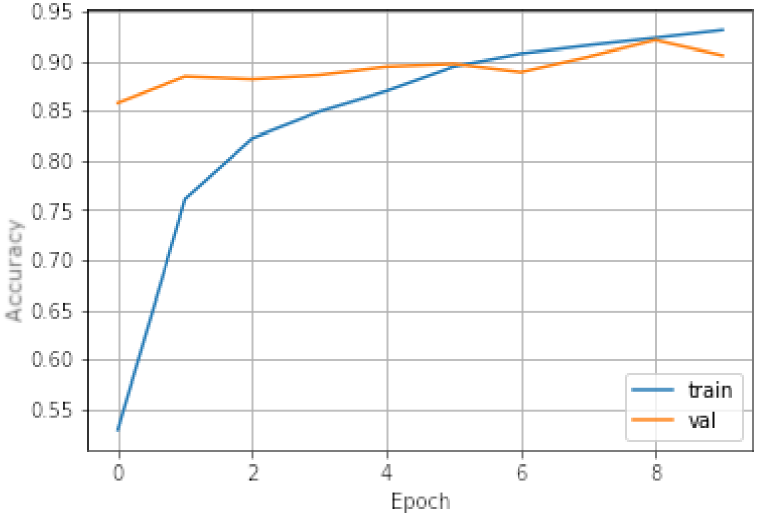
Training

- 20 epochs, categorical cross-entropy as loss function, Adam as optimizer
- Other metrics: accuracy
- Early stopping (patience on validation loss equal to 5) made training stop after 10 epochs.
- Training set
  - loss 0.20
  - accuracy 93.15%
- Validation set
  - loss 0.31
  - accuracy 90.59%



Training

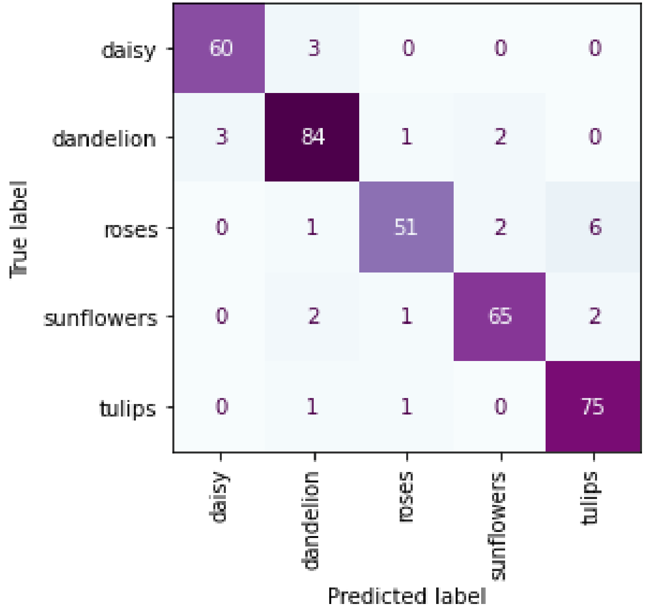
- 20 epochs, categorical cross-entropy as loss function, Adam as optimizer
- Other metrics: accuracy
- Early stopping (patience on validation loss equal to 5) made training stop after 10 epochs.
- Training set
  - loss 0.20
  - accuracy 93.15%
- Validation set
  - loss 0.31
  - accuracy 90.59%



Results on test set

- Only 25 images out of 360 misclassified.
- Test set
  - loss 0.20
  - accuracy 93.06%

	precision	recall	f1-score	support
daisy	0.95	0.95	0.95	63
dandelion	0.92	0.93	0.93	90
roses	0.94	0.85	0.89	60
sunflowers	0.94	0.93	0.94	70
tulips	0.90	0.97	0.94	77



Results on test set

True class	Predicted class	% daisy	% dandelion	% roses	% sunflowers	% tulips
daisy	daisy	99.09	0.50	0.05	0.18	0.18

True class	Predicted class	% daisy	% dandelion	% roses	% sunflowers	% tulips
roses	roses	< 0.01	< 0.01	> 99.99	< 0.01	< 0.01

Results on test set

True class	Predicted class	% daisy	% dandelion	% roses	% sunflowers	% tulips
daisy	sunflowers	21.84	5.69	1.90	69.40	1.17

True class	Predicted class	% daisy	% dandelion	% roses	% sunflowers	% tulips
dandelion	daisy	31.17	29.56	9.94	16.73	12.59

Data augmentation

- Transformations applied to training set:
  - Horizontal flip
  - Zoom (~20%)
  - Translations (~45 pixels in two of the four possible directions)
  - Contrast change (~30%)
- From 2525 images to 12625

Data augmentation

- Transformations applied to training set:
  - Horizontal flip
  - Zoom (~20%)
  - Translations (~45 pixels in two of the four possible directions)
  - Contrast change (~30%)
- From 2525 images to 12625

Architecture

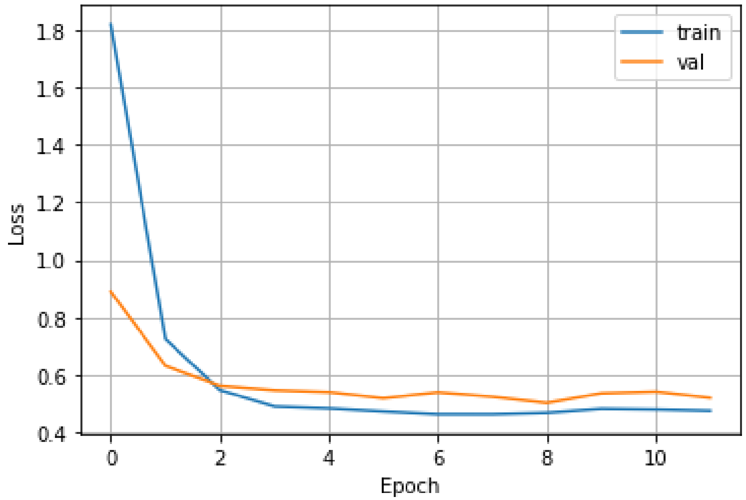
- Architecture similar to the one used during training without data augmentation
- Added L2-type regularization to the first two fully-connected layers in order to avoid overfitting

- Trainable parameters: ~1 million
- Non-trainable parameters: ~24 millions

```
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 224, 224, 3)]     0

tf.__operators__.getitem_1    (None, 224, 224, 3)      0
(SlicingOpLambda)

tf.nn.bias_add_1 (TFOpLambd   (None, 224, 224, 3)      0
a)

resnet50 (Functional)        (None, 7, 7, 2048)        23587712

global_average_pooling2d_1    (None, 2048)             0
(GlobalAveragePooling2D)

dense_3 (Dense)              (None, 512)               1049088

dropout_2 (Dropout)          (None, 512)               0

dense_4 (Dense)              (None, 128)               65664

dropout_3 (Dropout)          (None, 128)               0

dense_5 (Dense)              (None, 5)                 645
```

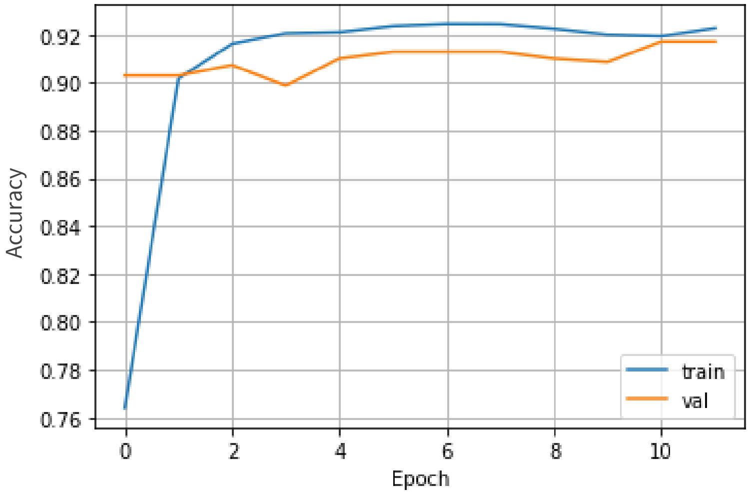
Training

- 20 epochs, categorical cross-entropy as loss function, Adam as optimizer
- Other metrics: accuracy
- Early stopping (patience on validation loss equal to 5) made training stop after 12 epochs.
- Training set
  - loss 0.48
  - accuracy 92.28%
- Validation set
  - loss 0.52
  - accuracy 91.71%



Training

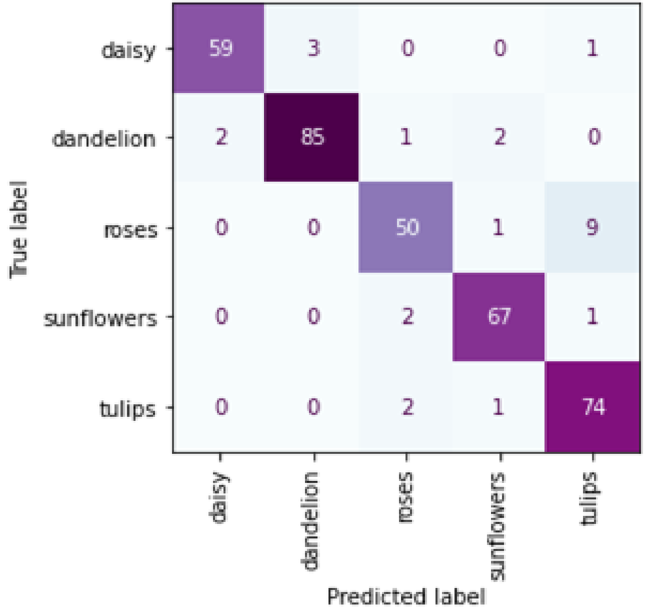
- 20 epochs, categorical cross-entropy as loss function, Adam as optimizer
- Other metrics: accuracy
- Early stopping (patience on validation loss equal to 5) made training stop after 12 epochs.
- Training set
  - loss 0.48
  - accuracy 92.28%
- Validation set
  - loss 0.52
  - accuracy 91.71%



Results on test set

- Only 25 images out of 360 misclassified.
- Test set
  - loss 0.45
  - accuracy 93.06%

	precision	recall	f1-score	support
daisy	0.97	0.94	0.95	63
dandelion	0.97	0.94	0.96	90
roses	0.91	0.83	0.87	60
sunflowers	0.94	0.96	0.95	70
tulips	0.87	0.96	0.91	77



Results on test set

True class	Predicted class	% daisy	% dandelion	% roses	% sunflowers	% tulips
daisy	daisy	98.97	0.53	0.09	0.13	0.37

True class	Predicted class	% daisy	% dandelion	% roses	% sunflowers	% tulips
roses	roses	< 0.01	< 0.01	99.87	< 0.01	0.13

Results on test set

True class	Predicted class	% daisy	% dandelion	% roses	% sunflowers	% tulips
daisy	daisy	88.22	2.18	0.98	7.31	1.31

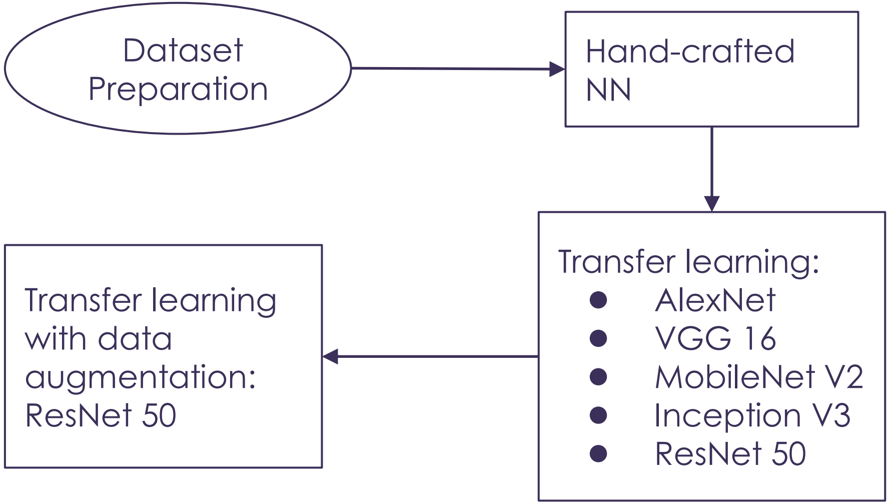
True class	Predicted class	% daisy	% dandelion	% roses	% sunflowers	% tulips
dandelion	daisy	47.87	31.06	3.94	10.14	6.99

Comparison ResNet 50

	Training set		Validation set		Test set	
	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
without data augmentation	0.20	93.15	0.31	90.59	0.20	93.06
with data augmentation	0.48	92.28	0.52	91.71	0.45	93.06

Conclusions

- Data cleaning
  - Remove images where the flower is not in the foreground
- Unknown class
- Finetuning



Backup

Links

- Google Drive folder: https://drive.google.com/drive/folders/1RKzAduIrG9x3HgBxDoPhbRsBXJMRC2NH?usp=sharing&authuser=2
- Dataset: http://download.tensorflow.org/example_images/flower_photos.tgz

Hand-crafted neural network

CODE

```
inputs = keras.Input((224,224,3))

x = keras.layers.Conv2D(32, 3, padding="same", kernel_regularizer=tf.keras.regularizers.l2(0.001))(inputs)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Activation("relu")(x)
x = keras.layers.MaxPooling2D(3, strides=3, padding="same")(x)

x = keras.layers.Conv2D(64, 3, padding="same", kernel_regularizer=tf.keras.regularizers.l2(0.001))(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Activation("relu")(x)
x = keras.layers.MaxPooling2D(3, strides=3, padding="same")(x)
x = keras.layers.Dropout(0.3)(x)

x = keras.layers.Conv2D(128, 3, padding="same", kernel_regularizer=tf.keras.regularizers.l2(0.001))(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Activation("relu")(x)
x = keras.layers.MaxPooling2D(3, strides=3, padding="same")(x)
x = keras.layers.Dropout(0.3)(x)

x = keras.layers.Conv2D(256, 3, padding="same", kernel_regularizer=tf.keras.regularizers.l2(0.001))(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Activation("relu")(x)
x = keras.layers.GlobalMaxPooling2D()(x)
x = keras.layers.Dropout(0.3)(x)

x = keras.layers.Dense(64, activation='relu')(x)
x = keras.layers.Dropout(0.1)(x)

outputs = keras.layers.Dense(num_classes, activation="softmax")(x)
model = keras.Model(inputs, outputs)
```

MobileNet V2

CODE

```
inputs = keras.Input(shape=(224, 224, 3))
x = tf.keras.applications.mobilenet_v2.preprocess_input(inputs)
x = mobile_net(x, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dense(1024, activation='relu')(x)

outputs = keras.layers.Dense(num_classes, activation='softmax')(x)

model = keras.Model(inputs, outputs)
```

AlexNet

CODE

```
AlexNet_model = torch.hub.load('pytorch/vision:v0.6.0', 'alexnet', pretrained = "imagenet")

AlexNet_model.classifier[4] = torch.nn.Linear(4096,2048)

AlexNet_model.classifier[6] = torch.nn.Linear(2048, NUMBER_CLASSES)
```

Inception V3

CODE

```
inputs = keras.Input(shape=(224, 224, 3))

x = tf.keras.applications.inception_v3.preprocess_input(inputs)

x = inception(x, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)

x = keras.layers.Dense(100, activation='relu')(x)
x = keras.layers.Dropout(0.5)(x)

x = keras.layers.Dense(50, activation='relu')(x)
x = keras.layers.Dropout(0.5)(x)

outputs = keras.layers.Dense(num_classes, activation='softmax')(x)

model = keras.Model(inputs, outputs)
```

VGG 16

CODE

```
inputs = keras.Input(shape=(224, 224, 3))
x = tf.keras.applications.vgg16.preprocess_input(inputs)

x = vgg(x, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)

x = keras.layers.Dense(100, activation='relu')(x)
x = keras.layers.Dropout(0.5)(x)

x = keras.layers.Dense(50, activation='relu')(x)
x = keras.layers.Dropout(0.5)(x)

outputs = keras.layers.Dense(num_classes, activation='softmax')(x)

model = keras.Model(inputs, outputs)
```

ResNet 50

CODE

```
inputs = keras.Input(shape=(224, 224, 3))

x = keras.applications.resnet50.preprocess_input(inputs)
x = resnet(x, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dense(100, activation='relu')(x)
x = keras.layers.Dropout(0.5)(x)
x = keras.layers.Dense(50, activation='relu')(x)
x = keras.layers.Dropout(0.5)(x)
outputs = keras.layers.Dense(5, activation='softmax')(x)

model = keras.Model(inputs, outputs)
```

ResNet 50 DA

CODE

```
inputs = keras.Input(shape=(224, 224, 3))

x = keras.applications.resnet50.preprocess_input(inputs)
x = resnet(x, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dense(100, activation='relu', kernel_regularizer=keras.regularizers.l2(0.01))(x)
x = keras.layers.Dropout(0.5)(x)
x = keras.layers.Dense(50, activation='relu', kernel_regularizer=keras.regularizers.l2(0.01))(x)
x = keras.layers.Dropout(0.5)(x)
outputs = keras.layers.Dense(5, activation='softmax')(x)

model = keras.Model(inputs, outputs)
```