AI Engineering Project

Final Presentation

Group 4 (Jungo, Luegmayer, Müllner & Schweitzer)

Project - Cats vs Dogs

Dataset

- TensorFlow Dataset - (cats_vs_dogs)
  - **25.000** Pictures in the dataset
  - **23.262** Valid pictures
  - **1.738** Corrupted images that are dropped

No pre trained Model

Model Architecture - Version 1

- **Conv2D Layers**
  - Extract hierarchical image features
- **MaxPooling Layers**
  - Reduce spatial dimensions, retain dominant features
- **Flatten Layer**
  - Converts 3D feature maps into 1D vector
- **Dense Layer**
  - Learns task-specific high-level representations
- **Dropout Layer**
  - Prevents overfitting during training
- **Output Layer**
  - Sigmoid activation for binary classification

Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 158, 158, 32)	896
max_pooling2d_9 (MaxPooling2D)	(None, 79, 79, 32)	0
conv2d_10 (Conv2D)	(None, 77, 77, 64)	18,496
max_pooling2d_10 (MaxPooling2D)	(None, 38, 38, 64)	0
conv2d_11 (Conv2D)	(None, 36, 36, 64)	36,928
max_pooling2d_11 (MaxPooling2D)	(None, 18, 18, 64)	0
flatten_3 (Flatten)	(None, 20736)	0
dense_6 (Dense)	(None, 64)	1,327,168
dropout_3 (Dropout)	(None, 64)	0
dense_7 (Dense)	(None, 1)	65

Total params: 1,383,553 (5.28 MB)
Trainable params: 1,383,553 (5.28 MB)
Non-trainable params: 0 (0.00 B)

Model Architecture - Version 2

- **Conv2D Layers + BatchNorm Layers**
  - Extract hierarchical image features, normalize activations
- **MaxPooling Layers**
  - Reduce spatial dimensions, retain dominant features
- **Flatten Layer**
  - Converts 3D feature maps into 1D vector
- **Dense Layer**
  - Learns task-specific high-level representations
- **Dropout Layer**
  - Prevents overfitting during training
- **Output Layer**
  - Sigmoid activation for binary classification

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 158, 158, 8)	224
batch_normalization (BatchNormalization)	(None, 158, 158, 8)	32
max_pooling2d (MaxPooling2D)	(None, 79, 79, 8)	0
conv2d_2 (Conv2D)	(None, 77, 77, 16)	1,168
batch_normalization_1 (BatchNormalization)	(None, 77, 77, 16)	64
max_pooling2d_1 (MaxPooling2D)	(None, 38, 38, 16)	0
conv2d_3 (Conv2D)	(None, 36, 36, 32)	4,640
batch_normalization_2 (BatchNormalization)	(None, 36, 36, 32)	128
max_pooling2d_2 (MaxPooling2D)	(None, 18, 18, 32)	0
flatten (Flatten)	(None, 10368)	0
dense (Dense)	(None, 32)	331,808
dropout (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 1)	33

Total params: 338,097 (1.29 MB)
Trainable params: 337,985 (1.29 MB)
Non-trainable params: 112 (448.00 B)

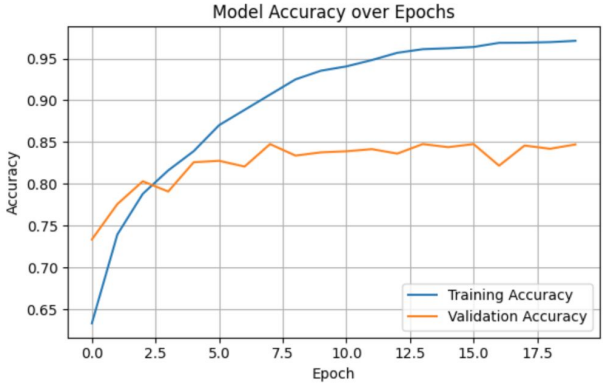
No pre trained Model (5)

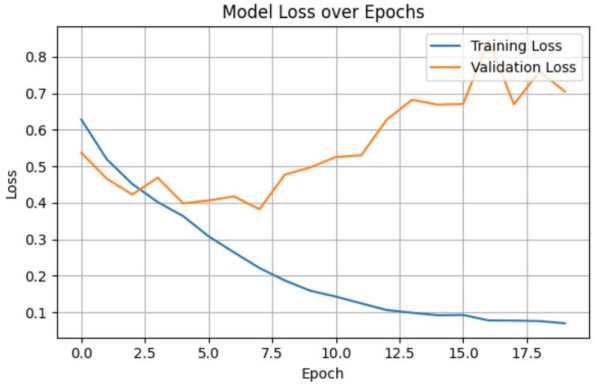
Model	Augmentation	BatchNorm	Conv Filters	Dense Units	Params	Train Acc	Val Acc	Overfitting	Notes
Base	No	No	32 → 64 → 64	64	~1.4M	97.12%	84.72%	High	Overfitting visible
Mod-1	No	No	8 → 16 → 32	32	~330k	94.01%	81.62%	Reduced	Simpler, still overfitting
Mod-2	No	No	4 → 8 → 16	16	~84k	75.66%	75.95%	None	Very balanced, efficient
Mod-3	Yes	No	4 → 8 → 16	16	~84k	71.78%	75.19%	None	Augmentation helps generalization
Mod-4	Yes	Yes	8 → 16 → 32	32	~332k	78.97%	79.90%	None	Best generalization and lowest loss

No pre trained Model - Result

Base

First Model

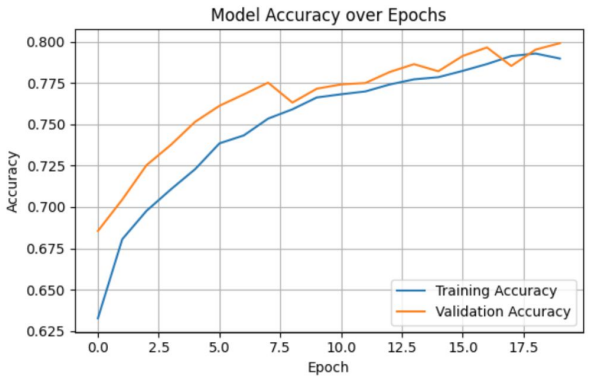


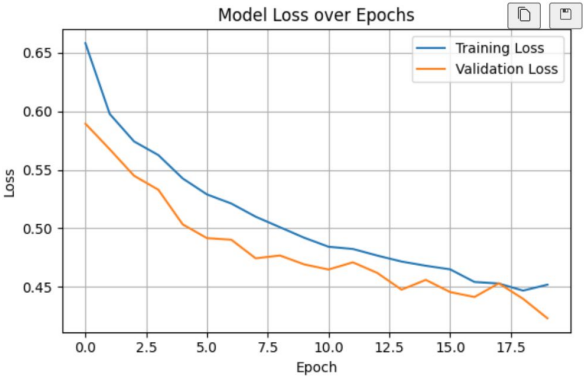



```
Final Training Metrics:
Training Accuracy: 0.9712
Validation Accuracy: 0.8472
Training Loss: 0.0703
Validation Loss: 0.7046
```

Modified 4

Last Model






```
Final Training Metrics:
Training Accuracy: 0.7897
Validation Accuracy: 0.7990
Training Loss: 0.4518
Validation Loss: 0.4230
```

Data Augmentation

Data Augmentation

Data Augmentation

```
# Data preprocessing
IMG_SIZE = 160
BATCH_SIZE = 32
AUTOTUNE = tf.data.AUTOTUNE

# Preprocessing (resize + normalize)
def preprocess(image, label):
    image = tf.cast(image, tf.float32)
    image = tf.image.resize(image, [IMG_SIZE, IMG_SIZE])
    image = image / 255.0
    return image, label

# Data-Augmentation pipeline
data_augmentation = tf.keras.Sequential([
    # Randomly crop somewhere between 60-100% of the image, then resize
    tf.keras.layers.RandomCrop(IMG_SIZE, IMG_SIZE),
    tf.keras.layers.RandomFlip("horizontal"),
    tf.keras.layers.RandomRotation(0.1),
    tf.keras.layers.RandomZoom(0.1, 0.1),
])
```

Pre trained Model

Model Architecture

- **MobileNetV2 Layer**:
  - Extracts rich feature maps from images
  - Uses pre-trained weights from ImageNet
- **Average Pooling Layer**:
  - Reduces the spatial dimensions
  - Single feature vector of 1280 units
- **Dense Layer**:
  - Learns task-specific representations
- **Dropout Layer**:
  - Prevent overfitting during training
- **Output Layer**:
  - Sigmoid activation function
  - Binary probability for classification

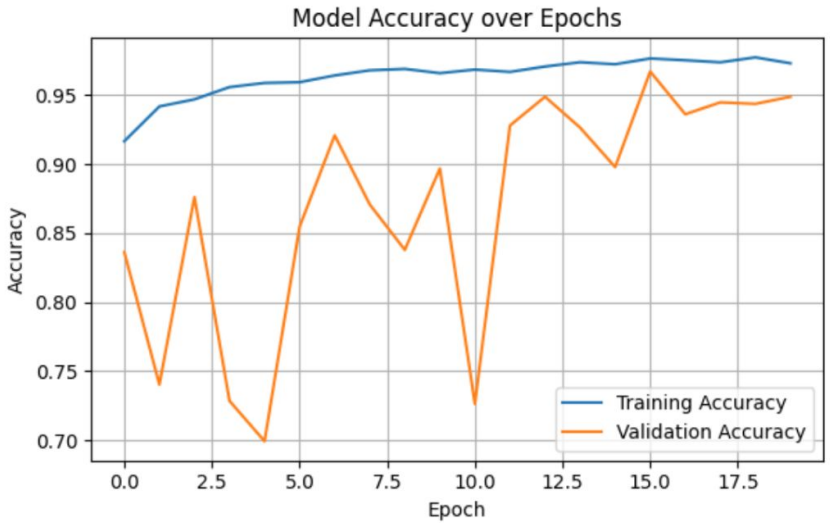
Layer (type)	Output Shape	Param #
mobilenetv2_1.00_160 (Functional)	(None, 5, 5, 1280)	2,257,984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
dense (Dense)	(None, 128)	163,968
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129

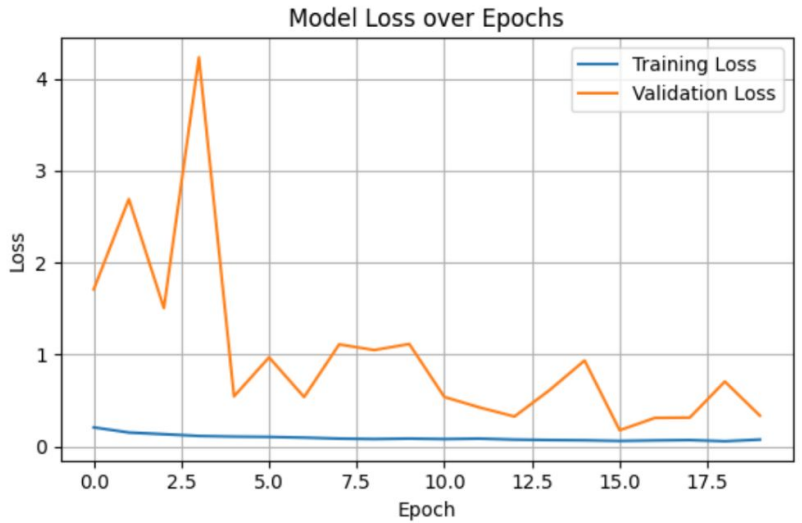
Total params: 2,422,081 (9.24 MB)
Trainable params: 2,387,969 (9.11 MB)
Non-trainable params: 34,112 (133.25 KB)

Pre trained Model

Model	Pretrained	Augmentation	BatchNorm	Conv Filters	Dense Units	Params	Train Acc	Val Acc	Val Loss	Notes
Base	No	No	No	32 → 64 → 64	64	~1.4M	97.12%	84.72%	0.7046	Strong overfitting
Mod-1	No	No	No	8 → 16 → 32	32	~330k	94.01%	81.62%	0.6774	Reduced capacity
Mod-2	No	No	No	4 → 8 → 16	16	~84k	75.66%	75.95%	0.4993	Lightweight, well-regularized
Mod-3	No	Yes	No	4 → 8 → 16	16	~84k	71.78%	75.19%	0.5266	Better generalization
Mod-4	No	Yes	Yes	8 → 16 → 32	32	~332k	78.97%	79.90%	0.4230	Best non-pretrained model
Pretrained	Yes	Yes	—	MobileNetV2	128	~2.4M	97.31%	**94.88%**	**0.3365**	Best overall

Pre trained Model

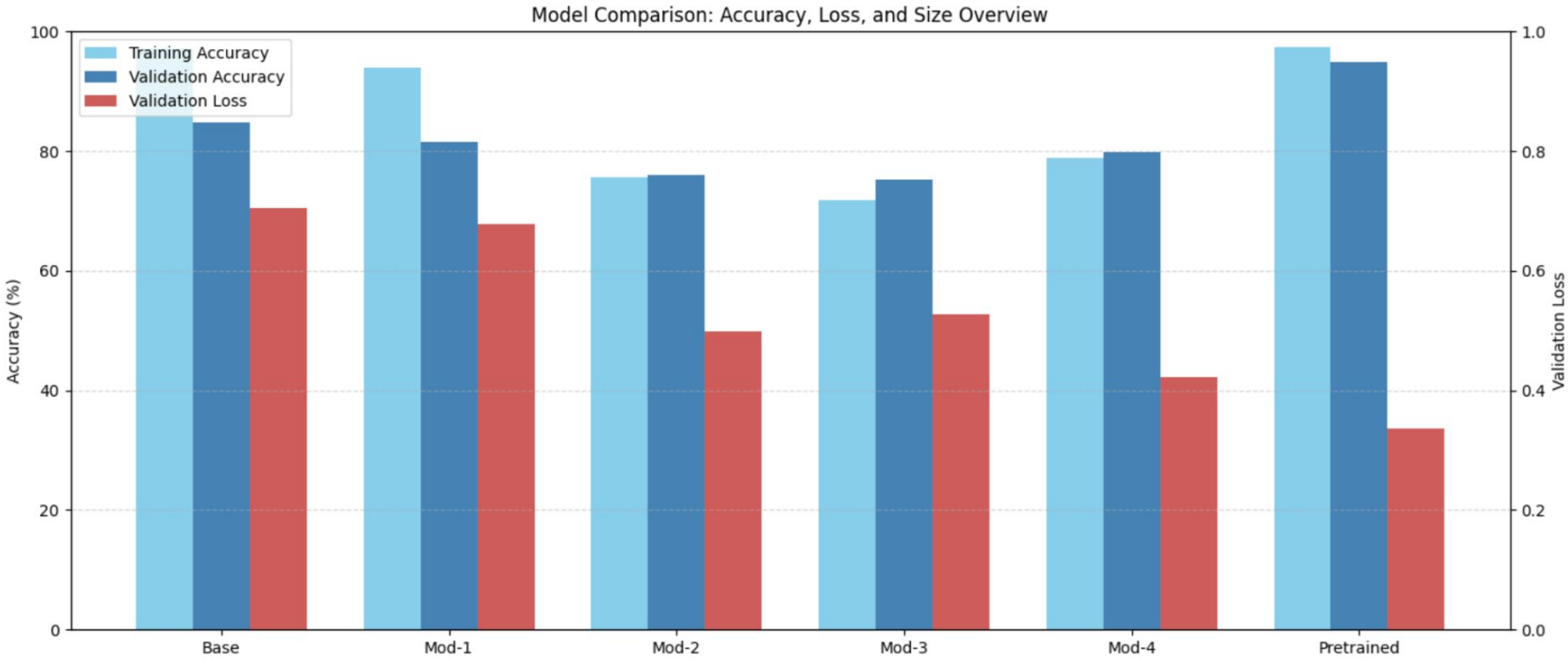




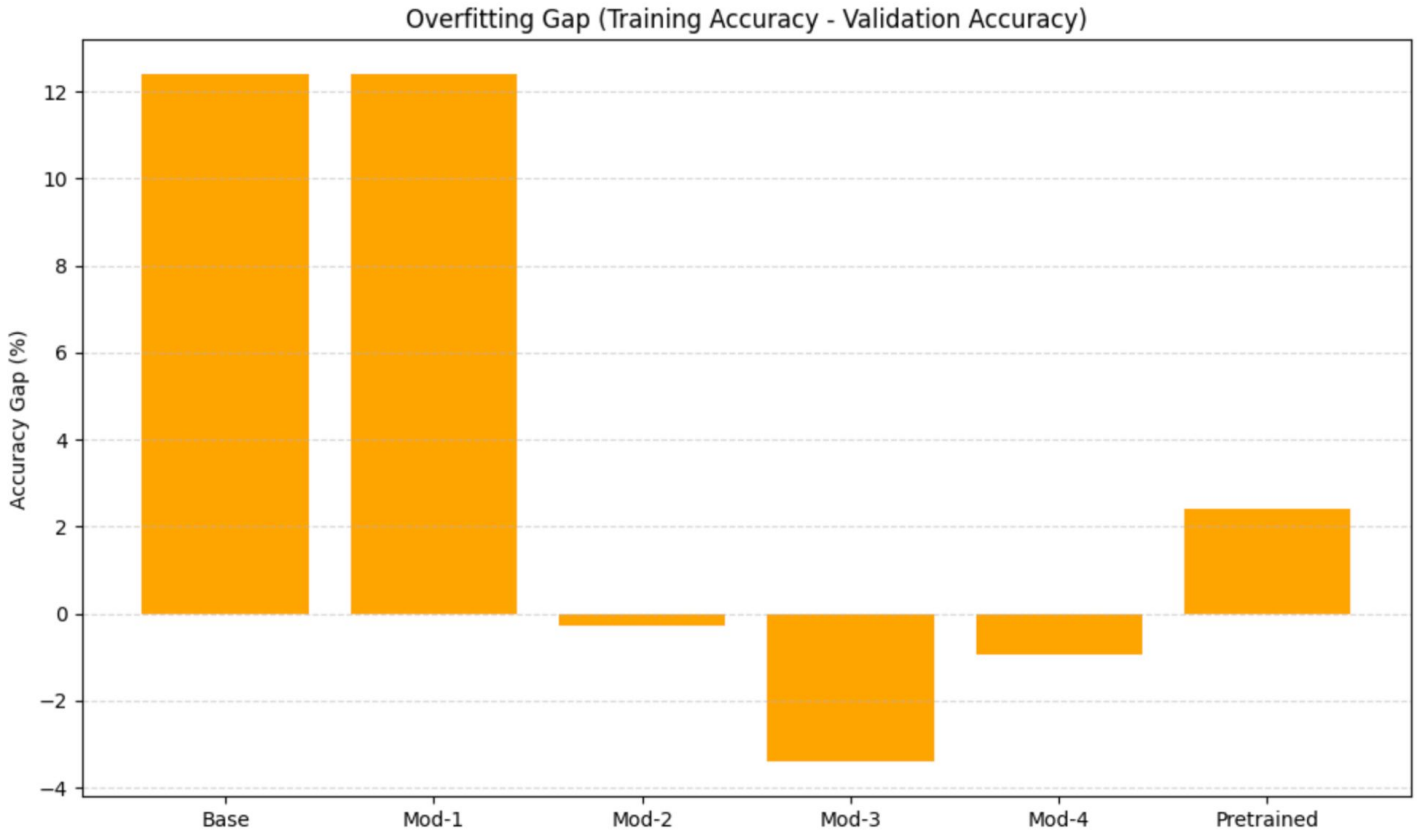

```
Final Training Metrics:
Training Accuracy: 0.9731
Validation Accuracy: 0.9488
Training Loss: 0.0761
Validation Loss: 0.3365
```

Overall Comparison

Overall Comparison



Overall Comparison



Overall Comparison