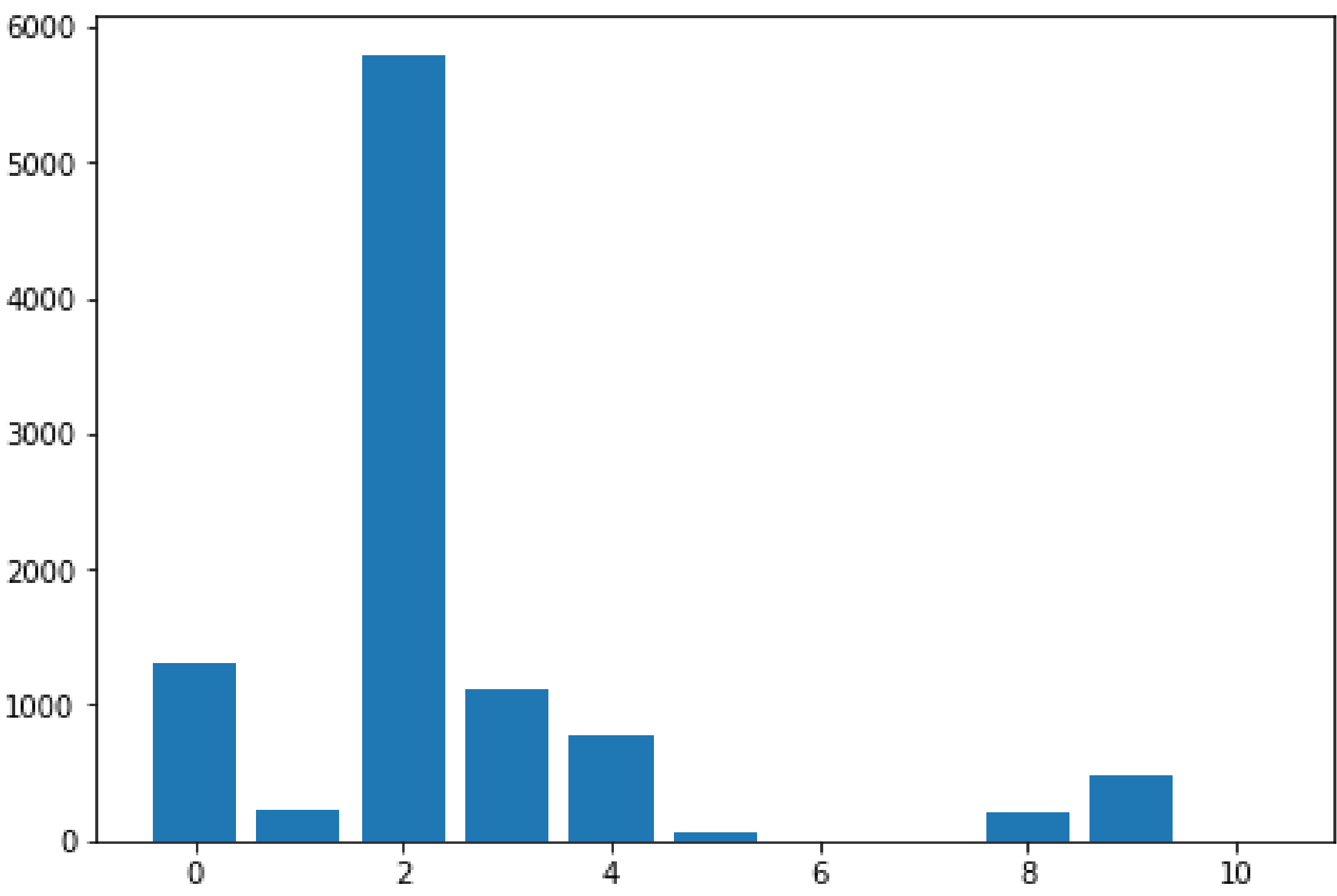


VM: Deep Learning mit Python

Case Study: Land Use and Land Cover

Data Access & Exploration

- Shape inspiziert: (10.000, 12, 33, 33, 6)
- Bilder visualisiert: 5 Patches über alle Monate
- Anzahl Trainingsdaten pro Klassen gezählt
- **Klasse 2 ist überrepräsentiert, während Klassen 6, 7 & 10 gar nicht vorhanden sind**



Data Preparation

- **Klassengewichte automatisch ermitteln**

```
# Calculate class weights automatically
class_weights_array = class_weight.compute_class_weight('balanced',
                                                         np.unique(y_train),
                                                         y_train)

# Create dictionary for the class weights
class_weights_dict = {
    0: class_weights_array[0],
    1: class_weights_array[1],
    2: class_weights_array[2],
    3: class_weights_array[3],
    4: class_weights_array[4],
    5: class_weights_array[5],
    6: 0,
    7: 0,
    8: class_weights_array[6],
    9: class_weights_array[7],
    10: 0,
}
```

- Data Augmentation des Trainings-Datensatzes
- In Test, Training & Validations Set aufgeteilt
- 1-hot Encoding

Choosing a Model

- Brauchen ein **3D CNN**, weil es 12 Bilder (Monate) pro Patch gibt
- Loss: (sparse) categorical crossentropy
- Optimierer: Adam
- Metrik: accuracy

Model: "sequential"		
Layer (type)	Output Shape	Param #

conv3d (Conv3D)	(None, 20, 31, 31, 32)	5216
max_pooling3d (MaxPooling3D)	(None, 5, 15, 15, 32)	0
batch_normalization (BatchNo	(None, 5, 15, 15, 32)	128

conv3d_1 (Conv3D)	(None, 3, 13, 13, 64)	55360
max_pooling3d_1 (MaxPooling3	(None, 1, 6, 6, 64)	0
batch_normalization_1 (Batch	(None, 1, 6, 6, 64)	256

flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 256)	590880

dense_1 (Dense)	(None, 128)	32896

dense_2 (Dense)	(None, 64)	8256

dense_3 (Dense)	(None, 32)	2080

dense_4 (Dense)	(None, 11)	363

Total params: 694,635		
Trainable params: 694,443		
Non-trainable params: 192		

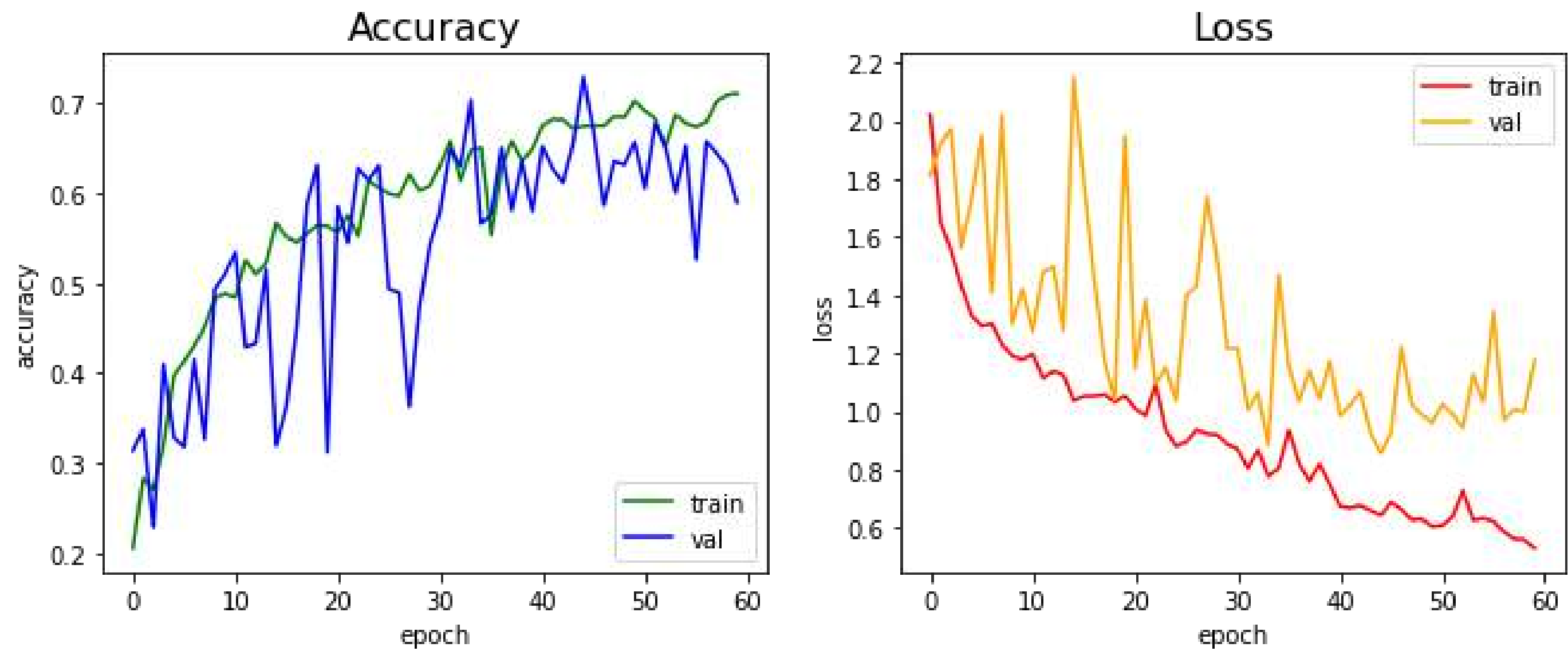
Alternative, die wir erstmal verworfen haben:
Transfer Learning mit vortrainiertem Modell:

- Problem: kein Modell mit vergleichlicher Input-Shape
- Workaround: zwei Modelle für jeweils drei Channel oder Training mit jeweils drei Channels nacheinander
→ Informationen gehen verloren

Training & Evaluation

- Training (fit):
- Mit automatisch ermittelten Klassengewichten
 - Callbacks: Early Stopping, Checkpoint
 - 60 Epochen

- Evaluation:
- **Accuracy: 0.6085 und Loss: 1.1794**



Prediction

- Zero-Padding: 16 Pixel mit Wert 0 um Bild herum
- **Naive Sliding Window:** Iteration über jeden Pixel einzeln
 - Verbesserte Lösungsansätze wie Convolutional Sliding Window Approach?
- Herausforderungen:
 - Dauert sehr lange, v.a. wenn man mit Numpy Arrays arbeitet beim Sliding Window Approach
 - Feine Elemente werden nicht perfekt erkannt
 - Randelemente durch Zero-Padding beeinflusst

