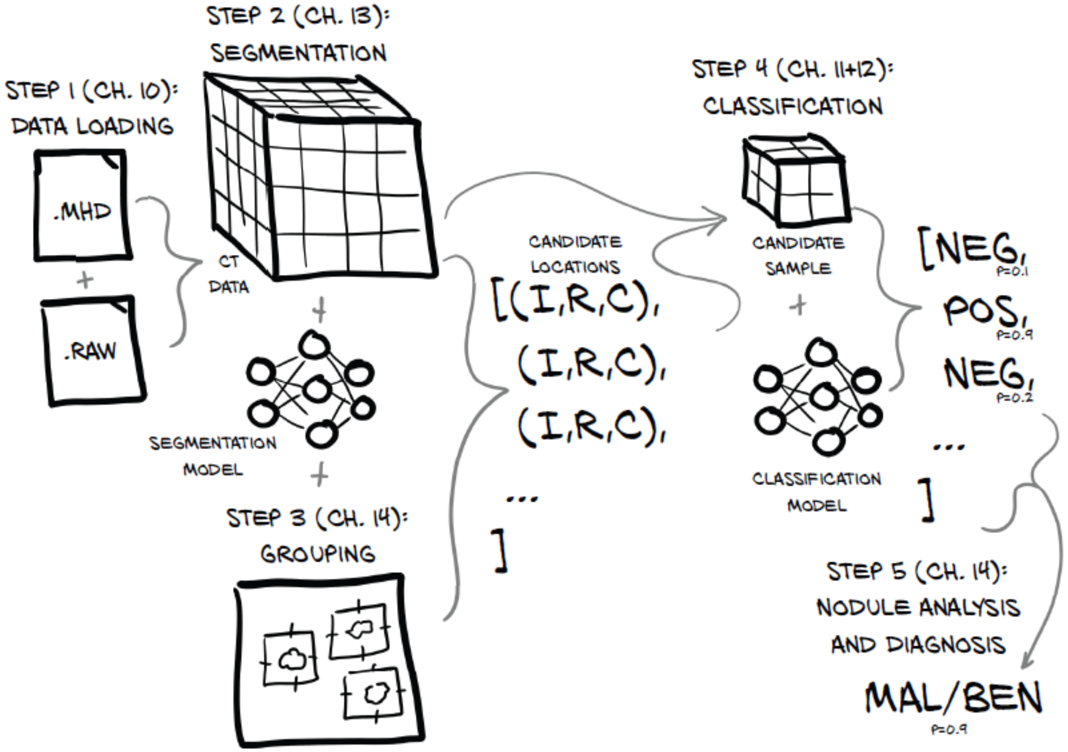
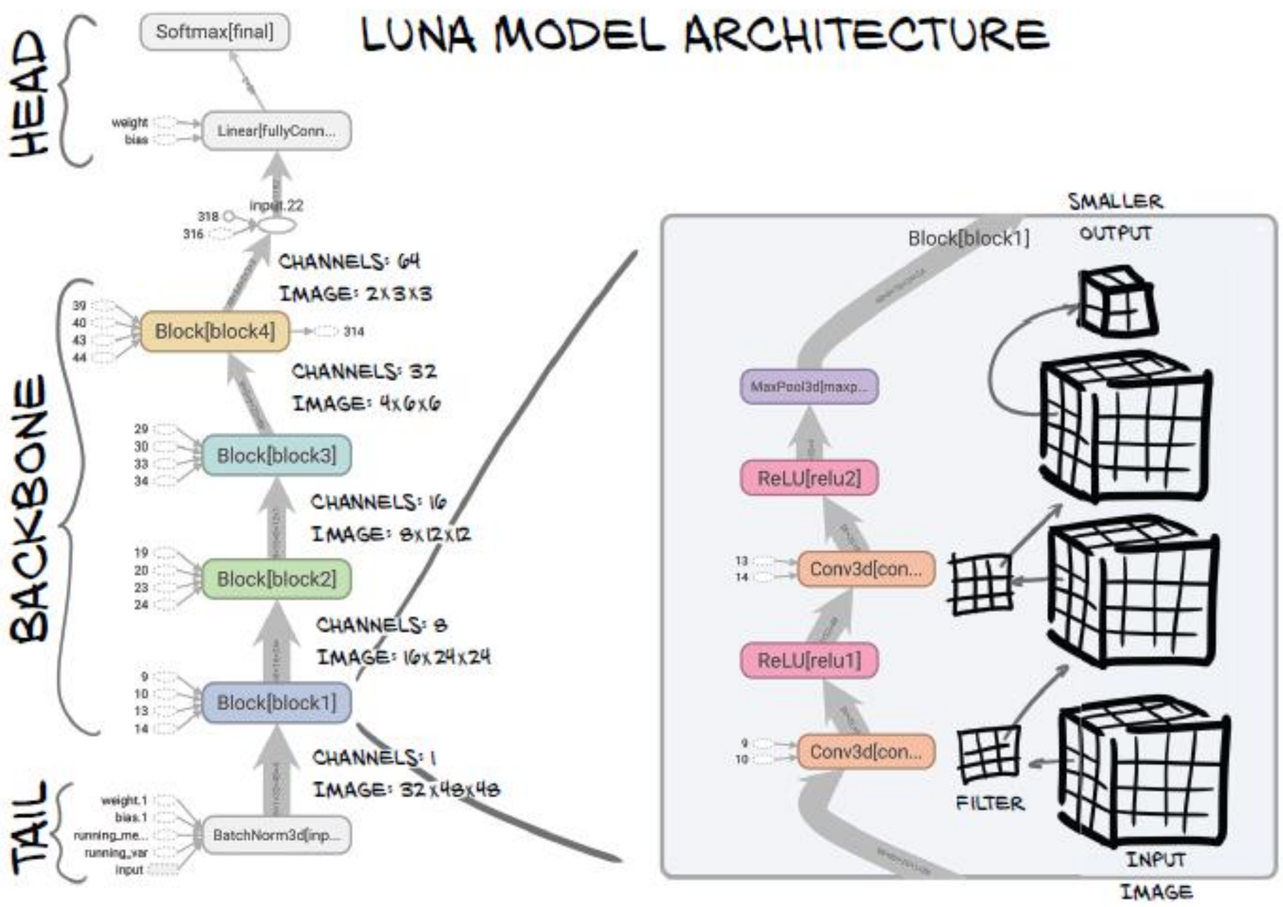
Vorlesung 9

CT Beispiel



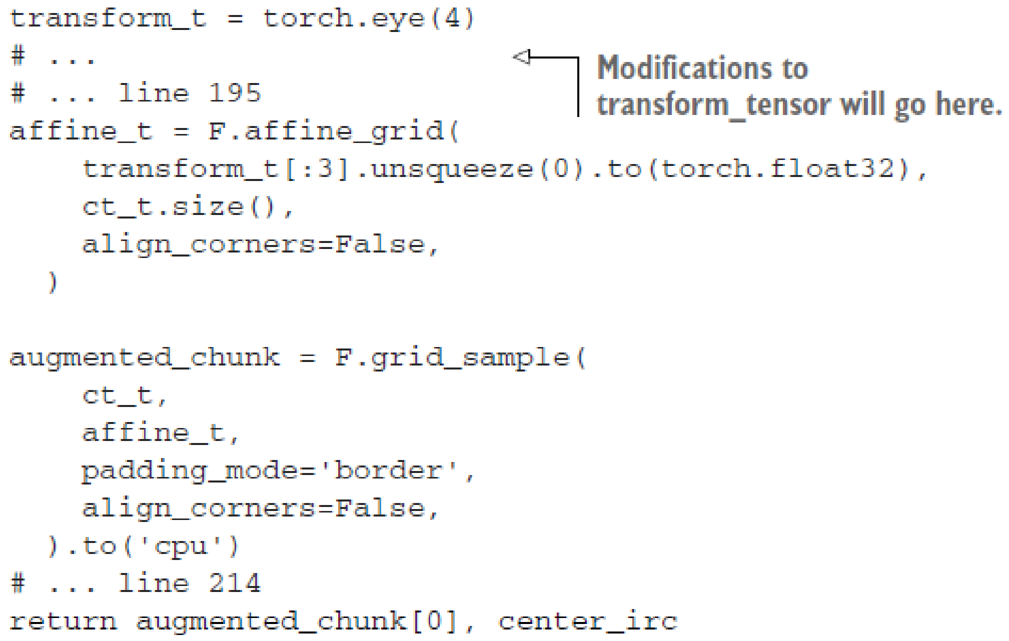
Quelle: Deep Learning with PyTorch

CT Beispiel



Quelle: Deep Learning with PyTorch

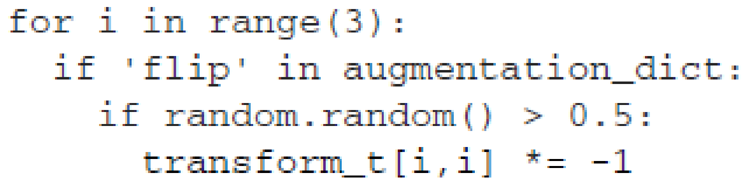
Mirroring



```
transform_t = torch.eye(4)
# ...
# ... line 195
affine_t = F.affine_grid(
    transform_t[:3].unsqueeze(0).to(torch.float32),
    ct_t.size(),
    align_corners=False,
  )

augmented_chunk = F.grid_sample(
    ct_t,
    affine_t,
    padding_mode='border',
    align_corners=False,
  ).to('cpu')
# ... line 214
return augmented_chunk[0], center_irc
```

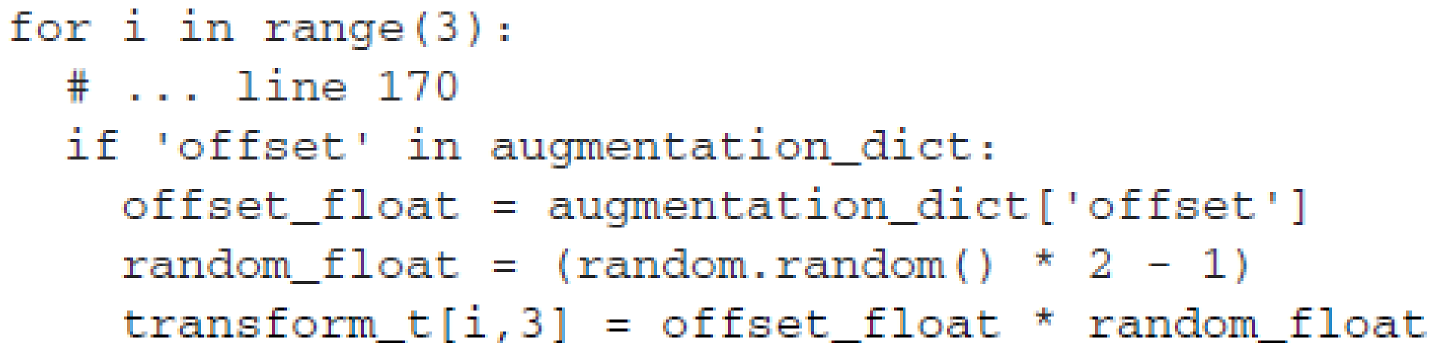
Modifications to transform_tensor will go here.



```
for i in range(3):
  if 'flip' in augmentation_dict:
    if random.random() > 0.5:
      transform_t[i,i] *= -1
```

Quelle: Deep Learning with PyTorch

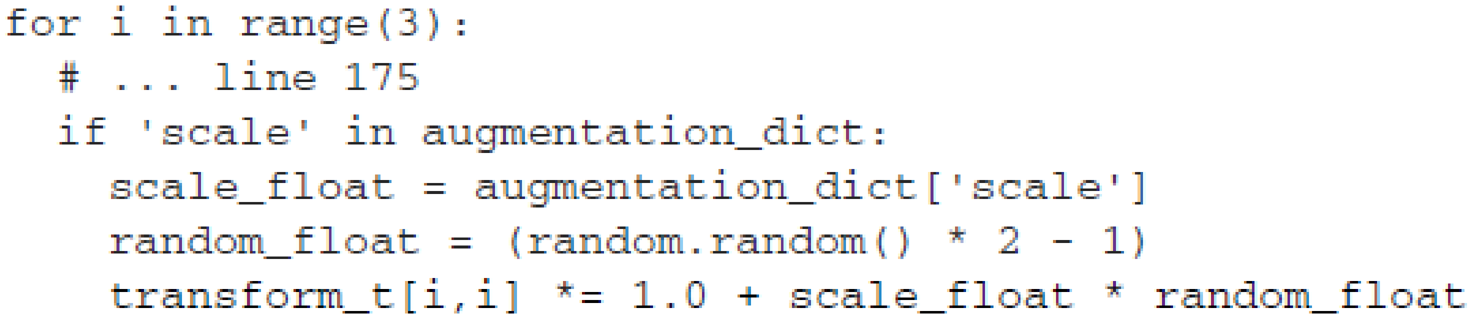
Shifting



```
for i in range(3):
  # ... line 170
  if 'offset' in augmentation_dict:
    offset_float = augmentation_dict['offset']
    random_float = (random.random() * 2 - 1)
    transform_t[i,3] = offset_float * random_float
```

Quelle: Deep Learning with PyTorch

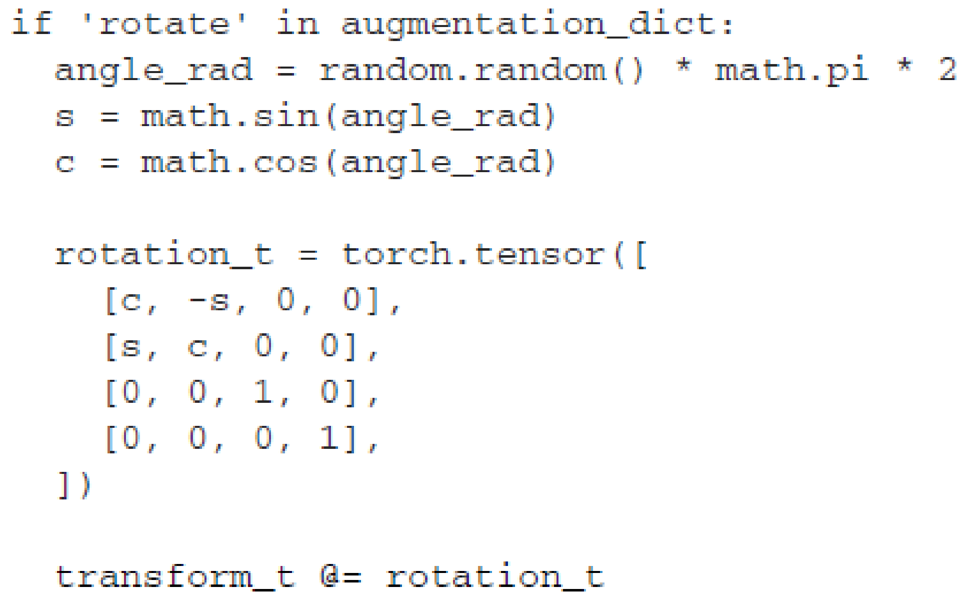
Scaling



```
for i in range(3):
  # ... line 175
  if 'scale' in augmentation_dict:
    scale_float = augmentation_dict['scale']
    random_float = (random.random() * 2 - 1)
    transform_t[i,i] *= 1.0 + scale_float * random_float
```

Quelle: Deep Learning with PyTorch

Rotating



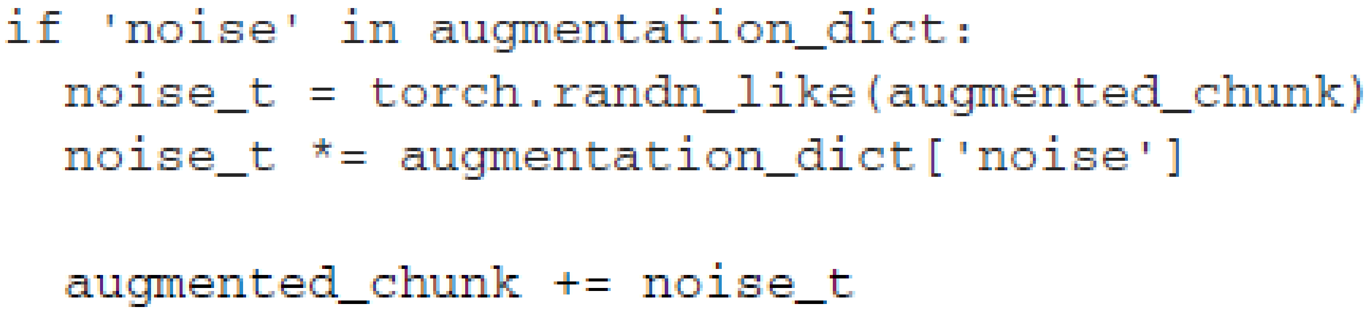
```
if 'rotate' in augmentation_dict:
  angle_rad = random.random() * math.pi * 2
  s = math.sin(angle_rad)
  c = math.cos(angle_rad)

  rotation_t = torch.tensor([
    [c, -s, 0, 0],
    [s, c, 0, 0],
    [0, 0, 1, 0],
    [0, 0, 0, 1],
  ])

  transform_t @= rotation_t
```

Quelle: Deep Learning with PyTorch

Noise

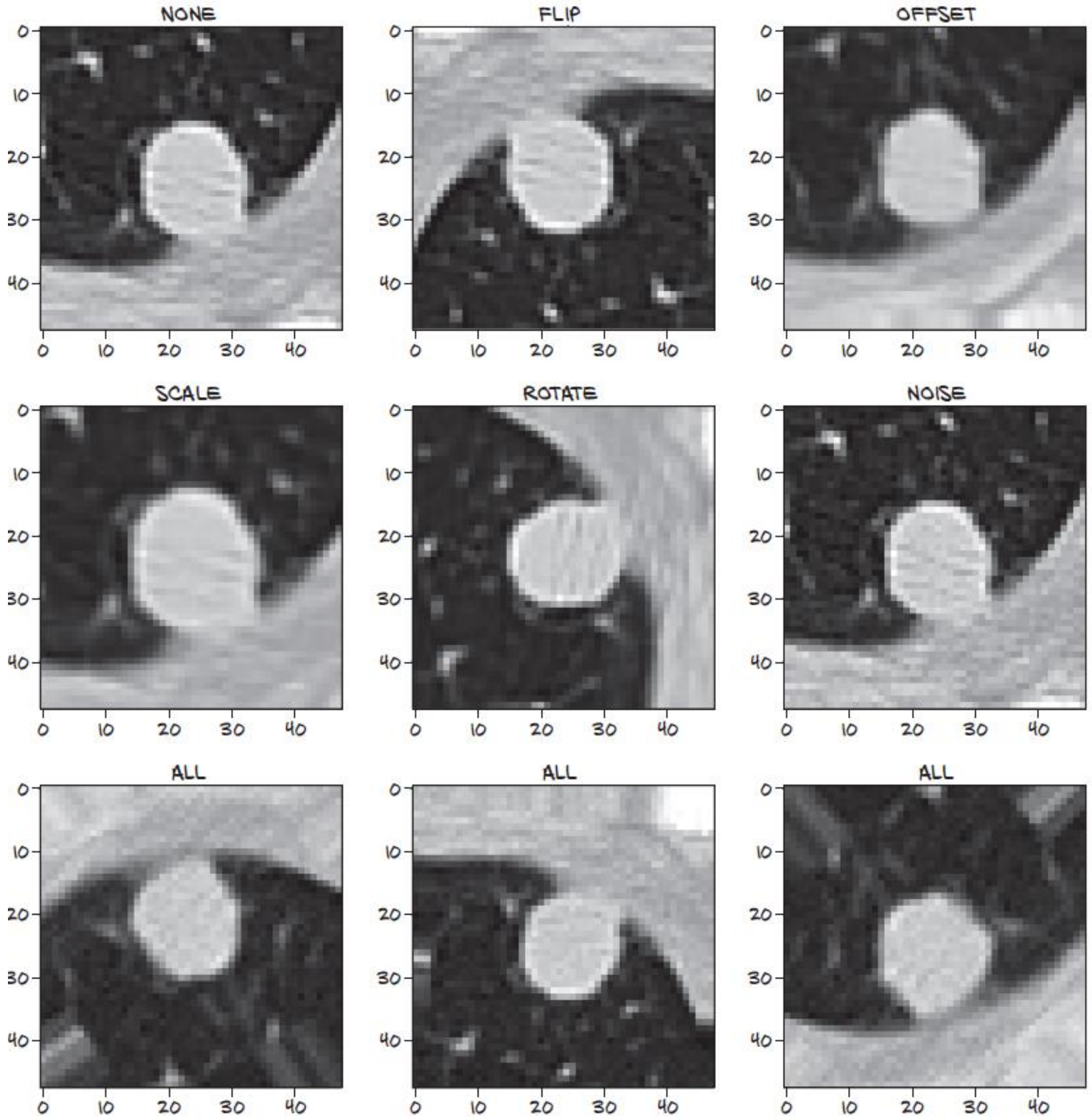


```
if 'noise' in augmentation_dict:
  noise_t = torch.randn_like(augmented_chunk)
  noise_t *= augmentation_dict['noise']

  augmented_chunk += noise_t
```

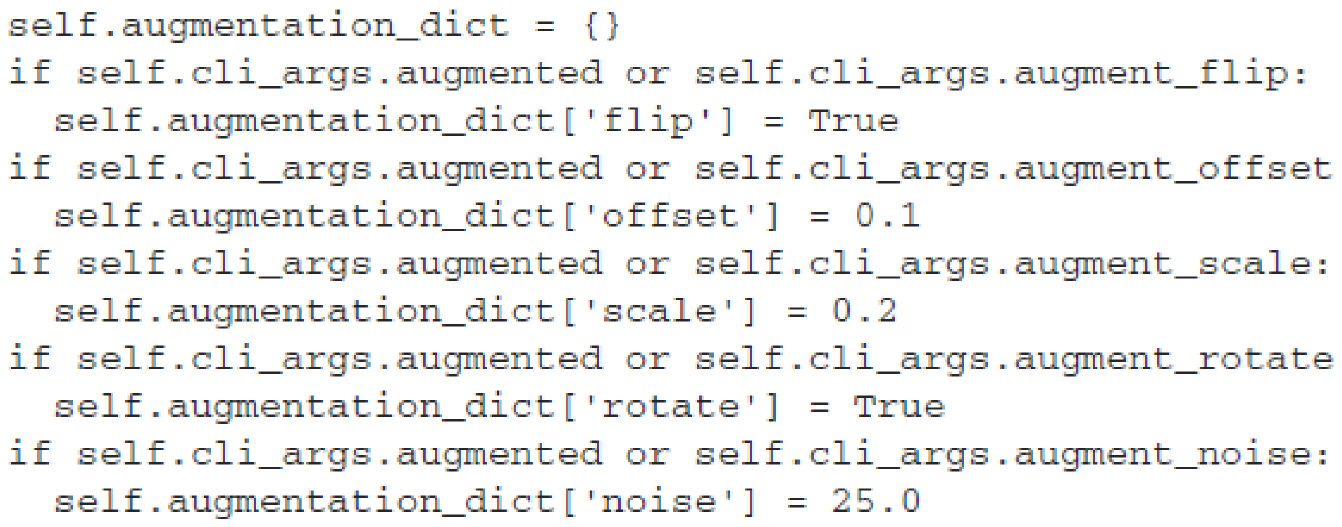
Quelle: Deep Learning with PyTorch

Augmentation Examples



Quelle: Deep Learning with PyTorch

Auswahl von Augmentation Methods



```
self.augmentation_dict = {}
if self.cli_args.augmented or self.cli_args.augment_flip:
  self.augmentation_dict['flip'] = True
if self.cli_args.augmented or self.cli_args.augment_offset
  self.augmentation_dict['offset'] = 0.1
if self.cli_args.augmented or self.cli_args.augment_scale:
  self.augmentation_dict['scale'] = 0.2
if self.cli_args.augmented or self.cli_args.augment_rotate
  self.augmentation_dict['rotate'] = True
if self.cli_args.augmented or self.cli_args.augment_noise:
  self.augmentation_dict['noise'] = 25.0
```

Quelle: Deep Learning with PyTorch

Aufgaben

- Modifizieren Sie das ResNet Beispiel aus dem Buch so, dass es dem Netzwerk aus dem Paper “Wide Residual Networks” von Zagoruyko et al. entspricht
- Verwenden Sie die gleichen Augmentation Methods wie in dem Paper (RandomHorizontalFlip und RandomCrop)
- Vergleichen Sie die Accuracy (Erkennungsraten) für Trainings- und Validierungsdaten mit dem Beispiel aus dem Buch
- Das Netzwerk dabei nur für die Klassifikation von Vögeln und Flugzeugen trainieren.
- Verwenden Sie 28 B(3,3) Blöcke mit k=2, [WRN-28-2-B(3,3)]

Abgabe per Github bis zum 20.06.2023 23:59 Uhr