**□ 모듈프로젝트 산출물**

1. 클라우드기반 데이터 보안 전문가 양성 과정 – 기반기술 인프라

|  |
| --- |
| 모듈프로젝트 4 |

**2021년 11월 9일**

**데이터 보안 전문가**

이름 : 정지원

1. **모듈프로젝트 내용**

* **비디오 데이터 분류**

|  |
| --- |
| AlexNet class  class AlexNet(Sequential):    def \_\_init\_\_(self, input\_shape, num\_classes):      super().\_\_init\_\_()      self.add(Conv2D(96, kernel\_size = (11, 11), strides = 4, padding = "valid", activation = "relu", input\_shape = input\_shape, kernel\_initializer = "he\_normal"))      self.add(MaxPooling2D(pool\_size = (3, 3), strides = (2, 2), padding = "valid", data\_format = "channels\_last"))      self.add(Conv2D(256, kernel\_size = (5, 5), strides = 1, padding = "same", activation = "relu", kernel\_initializer = "he\_normal"))      self.add(MaxPooling2D(pool\_size = (3, 3), strides = (2, 2), padding = "valid", data\_format = "channels\_last"))      self.add(Conv2D(384, kernel\_size = (3, 3), strides = 1, padding = "same", activation = "relu", kernel\_initializer = "he\_normal"))      self.add(Conv2D(384, kernel\_size = (3, 3), strides = 1, padding = "same", activation = "relu", kernel\_initializer = "he\_normal"))      self.add(Conv2D(256, kernel\_size = (3, 3), strides = 1, padding = "same", activation = "relu", kernel\_initializer = "he\_normal"))      self.add(MaxPooling2D(pool\_size = (3, 3), strides = (2, 2), padding = "valid", data\_format = "channels\_last"))      self.add(Flatten())      self.add(Dense(4096, activation = "relu"))      self.add(Dense(4096, activation = "relu"))      self.add(Dense(1000, activation = "relu"))      self.add(Dense(num\_classes, activation = "softmax")) |
| LeNet-5 class  class LeNet(Sequential):    def \_\_init\_\_(self, input\_shape, nb\_classes):      super().\_\_init\_\_()      self.add(Conv2D(6, kernel\_size = (5, 5), strides = (1, 1), activation = "relu", input\_shape = input\_shape, padding = "same"))      self.add(AveragePooling2D(pool\_size = (2, 2), strides = (2, 2), padding = "valid"))      self.add(Conv2D(16, kernel\_size = (5, 5), strides = (1, 1), activation = "relu", padding = "valid"))      self.add(AveragePooling2D(pool\_size = (2, 2), strides = (2, 2), padding = "valid"))      self.add(Flatten())      self.add(Dense(120, activation = "relu"))      self.add(Dense(84, activation = "relu"))      self.add(Dense(nb\_classes, activation = "softmax"))      self.compile(optimizer = "adam", loss = 'categorical\_crossentropy', metrics = ["accuracy"]) |
| Vision Transformer  # 패치 생성  class Patches(layers.Layer):      def \_\_init\_\_(self, patch\_size):          super(Patches, self).\_\_init\_\_()          self.patch\_size = patch\_size      def call(self, images):          batch\_size = tf.shape(images)[0]          patches = tf.image.extract\_patches(              images=images,              sizes=[1, self.patch\_size, self.patch\_size, 1],              strides=[1, self.patch\_size, self.patch\_size, 1],              rates=[1, 1, 1, 1],              padding="VALID",          )          patch\_dims = patches.shape[-1]          patches = tf.reshape(patches, [batch\_size, -1, patch\_dims])          return patches  class PatchEncoder(layers.Layer):      def \_\_init\_\_(self, num\_patches, projection\_dim):          super(PatchEncoder, self).\_\_init\_\_()          self.num\_patches = num\_patches          self.projection = layers.Dense(units=projection\_dim)          self.position\_embedding = layers.Embedding(              input\_dim=num\_patches, output\_dim=projection\_dim          )      def call(self, patch):          positions = tf.range(start=0, limit=self.num\_patches, delta=1)          encoded = self.projection(patch) + self.position\_embedding(positions)          return encoded  # 다층 퍼셉트론 구현 : MulitLayer Perceptron  def mlp(x, hidden\_units, dropout\_rate):      for units in hidden\_units:          x = layers.Dense(units, activation=tf.nn.gelu)(x)          x = layers.Dropout(dropout\_rate)(x)      return x  # 모델 정의  def create\_vit\_classifier():      inputs = layers.Input(shape=input\_shape)      augmented = data\_augmentation(inputs)      patches = Patches(patch\_size)(augmented)      encoded\_patches = PatchEncoder(num\_patches, projection\_dim)(patches)      for \_ in range(transformer\_layers):          x1 = layers.LayerNormalization(epsilon=1e-6)(encoded\_patches)          attention\_output = layers.MultiHeadAttention(              num\_heads=num\_heads, key\_dim=projection\_dim, dropout=0.1          )(x1, x1)          x2 = layers.Add()([attention\_output, encoded\_patches])          x3 = layers.LayerNormalization(epsilon=1e-6)(x2)          x3 = mlp(x3, hidden\_units=transformer\_units, dropout\_rate=0.1)          encoded\_patches = layers.Add()([x3, x2])      representation = layers.LayerNormalization(epsilon=1e-6)(encoded\_patches)      representation = layers.Flatten()(representation)      representation = layers.Dropout(0.5)(representation)      features = mlp(representation, hidden\_units=mlp\_head\_units, dropout\_rate=0.5)      logits = layers.Dense(num\_classes)(features)      model = keras.Model(inputs=inputs, outputs=logits)      return model  # 모델 실행  def run\_experiment(model):      optimizer = tfa.optimizers.AdamW(          learning\_rate=learning\_rate, weight\_decay=weight\_decay      )      model.compile(          optimizer=optimizer,          loss=keras.losses.SparseCategoricalCrossentropy(from\_logits=True),          metrics=[              keras.metrics.SparseCategoricalAccuracy(name="accuracy"),              keras.metrics.SparseTopKCategoricalAccuracy(5, name="top-5-accuracy"),          ],      )      checkpoint\_filepath = "/content/checkpoint"      checkpoint\_callback = keras.callbacks.ModelCheckpoint(          checkpoint\_filepath,          monitor="val\_accuracy",          save\_best\_only=True,          save\_weights\_only=True,      )      history = model.fit(          x=x\_train,          y=y\_train,          batch\_size=batch\_size,          epochs=num\_epochs,          validation\_split=0.1,          callbacks=[checkpoint\_callback],      )      model.load\_weights(checkpoint\_filepath)      \_, accuracy, top\_5\_accuracy = model.evaluate(x\_test, y\_test)      print(f"Test accuracy: {round(accuracy \* 100, 2)}%")      print(f"Test top 5 accuracy: {round(top\_5\_accuracy \* 100, 2)}%")      return history  vit\_classifier = create\_vit\_classifier()  history = run\_experiment(vit\_classifier) |
| https://lh6.googleusercontent.com/J2Tq99S2VL0e_165v5iQf1eMBpPyeqILdMfj40jnMz2KduOTkukujqRarBpo6r2oMFJEj1G9DEqpw8-4Shzg5_Y1KHSe3RYRJItqD1LWoPyz2Rk5feErtFgz9GMW7XFOWGXeX2nldnU |

|  |  |
| --- | --- |
| 성명 | 프로젝트 후 소감 |
| 김남준 | 이번 프로젝트는 다양한 모델을 이용하여 비디오 데이터를 분류하는 것이 목표였다. CNN이나 RNN같은 기본적인 모델부터 GoogLeNet이나 비전 트랜스포머 모델 같은 최신의 모델을 두루 접할 수 있는 기회였다. 프로젝트를 진행하면서 딥러닝 모델이 어떻게 발전하고 있는지 알 수 있었고, 그에 따른 컴퓨팅 파워가 중요하다는 것도 알게 되었다. 비전 분야의 경우 모델을 직접 만드는 경우보다 불러오는 경우가 훨씬 많기 때문에 앞으로 사용할 모델을 경험해볼 수 있는 좋은 공부였다고 생각한다. |
| 민병석 | 이번 프로젝트를 통해 다양한 모델들을 알 수 있게 되었습니다. 또 그 모델들의 특징, 역사 그리고 장, 단점을 한 번에 정리할 수 있는 좋은 기회였습니다. 기존 수업에서 배우지 않았던 다른 모델들을 활용하여 비디오를 분류하고, 그 모델들의 작동원리를 이해하며 비교할 수 있는 흥미로운 시간이었습니다. 이번 계기로 좀 더 다양한 모델들을 조사해 보고 비교해 볼 생각입니다.  그리고 각 모델을 이용해 어떻게 활용되고 어떻게 동작하는지 알 수 있게 되었습니다. |
| 배상현 | 교육 과정에 배운 합성곱 신경망과 순환 신경망을 이용해 비디오 데이터를 분류하는 프로젝트를 진행하였다. CNN과 LSTM 모델에 맞게 데이터를 전처리 하는 과정을 익힐 수 있었고, 여러 알고리즘 모델들이 어떤 장단점을 가지고 있어 어떤 경우에 이용하는 것이 좋은 지와 딥러닝 모델의 발전 과정을 알 수 있었다. 프로젝트를 수행하면서 특히 GoogleNet의 경우와 같이 컴퓨팅 리소스의 한계로 곤란을 겪는 경우가 있었는데, 이 부분을 보완하는 모델인 MobileNet을 보면서 알고리즘 경량화 중요성을 직접 확인하게 되었다. 조사한 모델들에 더해 앞으로도 딥러닝 기술 동향을 계속 주시하면서 따로 정리 해두어야겠다. |
| 신종유 | 프로젝트를 수행하면서 알고리즘에 대해 심층적으로 공부할 수 있었다. 사실 학교에서 컴퓨터 비전에 대해 배우긴 했지만 작동 원리에 그쳤다. 또한 우리 교육 과정에서도 실습을 하며 공부하였지만, 왜 주어진 모델을 써야 하는지, 모델의 역할과 함수의 의미가 잘 이해되지 않았다. 하지만 프로젝트에서 여러 모델을 공부하며 기존 모델의 한계와 새로운 해결책을 알게 되면서 CNN, LSTM에 대한 이해를 할 수 있게 되었다. 등장 배경 및 모델의 개발 역사가 중요하다는 것을 느꼈다. |
| 정지원 | 안녕하세요. SK쉴더스 AI 4기 교육생 정지원 입니다.  4차 프로젝트는 최종프로젝트 주제 선정 기간과 겹쳐서 1~3차 때 보다 더 정신이 없었던 것 같습니다. 4차 프로젝트는 수업시간에 배웠던 딥러닝 CNN, LSTM에 대한 연장선이었습니다. 많이 생소했던 분야였지만 수업시간 외에도 배우지 않았던 다른 모델들을 이용해서 분류 학습하고, 기존 모델들과 비교해 볼 수 있는 유익했던 시간이었습니다.  진도가 나갈 때는 수업내용 복습이 전부였지만 프로젝트를 하면서 책을 통해 새로운 알고리즘들도 공부하게 되었고, 배웠던 알고리즘들도 더 자세하게 학습할 수 있었습니다. 전체적으로 놓쳤던 부분들을 복습할 수 있는 좋은 기회였습니다  마지막 평가 프로젝트라 후회가 남지 않도록 잘 마무리 해야겠다고 생각했습니다.  제 자신을 봤을 때 아쉬운 부분들도 많았지만 팀원들이 그 부분들을 채워주었고,  마지막 프로젝트까지 좋은 사람들과 함께 할 수 있음에 감사했습니다. |