좋습니다. 아래는 **Temporal Fusion Transformer (TFT)** 모델을 소개하고 설명하는 데 사용할 수 있는 긴 대본입니다.

강의, 프레젠테이션, 영상 내레이션 등에 적합하며, 기술적이면서도 단계적으로 설명하는 구성을 따릅니다:

**🎙️ Temporal Fusion Transformer (TFT) 설명 대본**

안녕하세요. 오늘은 시계열 예측 분야에서 주목받는 딥러닝 모델인 **Temporal Fusion Transformer**, 줄여서 **TFT**에 대해 자세히 알아보겠습니다.

**1. 도입 – 왜 TFT인가?**

전통적인 시계열 예측 모델들은 제한된 정보만을 활용해 미래를 예측합니다.

예를 들어 ARIMA나 LSTM 기반 모델들은 보통 단일 입력을 중심으로 작동하거나, 복잡한 외부 변수들을 효과적으로 통합하기 어렵습니다.

하지만 실제 환경에서는 수많은 요인들이 시계열 데이터에 영향을 미칩니다.

예측 모델이 과거 정보뿐만 아니라 **미래에 이미 알고 있는 변수**, **범주형 정보**, 그리고 **불확실성**까지 반영할 수 있다면 어떨까요?

이 문제를 해결하기 위해 등장한 것이 바로 **Temporal Fusion Transformer**, 즉 TFT입니다.

TFT는 시계열 데이터를 예측하는 데 있어 다음의 핵심 기능들을 제공합니다:

* **과거 및 미래 입력을 모두 통합**
* **정적 변수, 범주형 변수, 연속형 변수 통합 처리**
* **가중치 해석 가능 (interpretability)**
* **예측 불확실성 추정 (Quantile Forecasting)**
* 그리고 **Transformer 구조 기반의 장기 의존성 학습 능력**

**2. 아키텍처 구조**

TFT는 다양한 모듈을 결합하여 구성된 복합적인 모델입니다. 주요 구성 요소는 다음과 같습니다:

**📌 2-1. Variable Selection Networks (VSNs)**

입력되는 여러 변수들 중, 어떤 변수가 예측에 중요한지를 판단하여 선택적으로 반영합니다.

이는 각 시점마다 동적으로 작동합니다.

정적 변수(static), 과거 시점 입력(past inputs), 미래 시점 입력(future inputs)에 각각 별도의 VSN이 존재합니다.

**📌 2-2. Gated Residual Networks (GRNs)**

GRN은 TFT의 핵심 모듈입니다.

입력값을 비선형적으로 변환하고, 이를 잔차 연결(residual connection)을 통해 원본 정보와 합칩니다.

GRN은 변수 선택 네트워크, 시계열 처리 모듈 등 TFT의 다양한 지점에서 사용됩니다.

**📌 2-3. LSTM Encoding**

Transformer에 입력되기 전에, 시계열 정보는 LSTM을 통해 한 번 인코딩됩니다.

이는 **Transformer가 시계열 초기 정보를 잘 받아들이게 하기 위한 전처리**라고 생각하면 됩니다.

**📌 2-4. Self-Attention 기반 Temporal Fusion Layer**

이제 본격적으로 Transformer 구조가 들어옵니다.

Transformer는 Attention 메커니즘을 통해 **장기 의존성을 학습**하고, **미래 예측에 중요한 시점들을 자동으로 강조**합니다.

이 부분이 LSTM 기반 모델과 차별되는 TFT의 강점입니다.

**📌 2-5. Quantile Loss를 통한 예측 불확실성 추정**

TFT는 단일 예측값이 아닌, **Quantile Loss**를 사용해 \*\*다수의 분위수(quantiles)\*\*를 예측합니다.

즉, 예측 결과는 단일 수치가 아닌 **범위 또는 분포**로 표현됩니다.

이는 비즈니스에서 **리스크 관리나 시나리오 분석**에 유용하게 활용될 수 있습니다.

**3. 입력 데이터 구성**

TFT를 사용하려면 데이터를 다음과 같은 형태로 구성해야 합니다:

* **time\_idx**: 시계열의 시간 인덱스
* **target**: 예측할 대상 변수
* **group\_id**: 시계열 그룹 (예: 국가별, 상품별 등)
* **known future variables**: 예측 시점에도 미리 아는 변수들 (예: 요일, 휴일 등)
* **observed past variables**: 과거에만 관측 가능한 변수들
* **static variables**: 변하지 않는 특성 (예: 국가명, 제품군 등)

pytorch-forecasting 패키지에서는 TimeSeriesDataSet 클래스를 통해 이 복잡한 입력 구성을 손쉽게 처리할 수 있습니다.

**4. 학습과 예측**

TFT 모델은 PyTorch Lightning과 호환되어 쉽게 학습 및 추론이 가능합니다:  
예측은 간단히 .predict() 메서드를 통해 수행할 수 있습니다.

**5. 해석 가능성 (Interpretability)**

TFT는 예측 결과뿐만 아니라 다음과 같은 해석 정보를 제공합니다:

* 변수 중요도 (variable importance)
* 각 시점에서의 attention 가중치
* 예측 구간 (prediction interval)

따라서 “왜 이런 결과가 나왔는가?“에 대한 설명을 제공할 수 있어, 실무 적용에 매우 유리합니다.

**6. 마무리**

Temporal Fusion Transformer는 시계열 예측을 위한 **가장 진보된 딥러닝 구조 중 하나**입니다.

복잡한 입력을 처리하고, 불확실성을 추정하며, 해석 가능한 결과를 제공하기 때문에 금융, 수요예측, 에너지, 헬스케어 등 다양한 산업에서 널리 활용되고 있습니다.

TFT는 단순한 LSTM이나 ARIMA를 넘어선, **진짜 시계열의 종합 모델**이라 할 수 있습니다.

필요하다면 이 내용을 슬라이드용으로 재구성하거나, 더 기술적인 수식 중심의 버전으로도 바꿔드릴 수 있습니다. 원하시면 말씀해 주세요.

안녕하세요, 오늘은 **딥러닝에서 자연어 처리의 판도를 바꾼 모델**, 바로 **Transformer**에 대해 알아보겠습니다.

Transformer는 2017년 Google Brain 팀의 논문 \*\*“Attention Is All You Need”\*\*에서 처음 소개되었습니다.

이 모델은 RNN이나 CNN 없이 **전적으로 Attention 메커니즘**에만 의존하여, **더 빠르고 병렬화 가능한 구조**를 제공한다는 점에서 큰 혁신이었습니다.

## 1. 왜 Transformer가 나왔는가?

기존의 RNN이나 LSTM 모델은 **순차적으로 데이터를 처리**해야 하기 때문에, 병렬화가 어렵고 긴 문장에서 **장기 의존성(long-term dependency)** 문제를 겪었습니다.

Transformer는 이 문제를 해결하기 위해 고안되었습니다.

바로 **Self-Attention**이라는 메커니즘을 통해, **모든 입력 단어가 서로를 동시에 참고**할 수 있도록 설계된 것이죠.

## 2. Transformer의 기본 구조

Transformer는 크게 **Encoder-Decoder 구조**로 이루어져 있습니다.

이 구조는 다음과 같은 세 가지 주요 블록으로 구성됩니다:

1. **Encoder**: 입력 문장을 벡터로 인코딩합니다. 여러 개의 encoder layer로 이루어져 있습니다.
2. **Decoder**: 이 벡터로부터 출력 문장을 생성합니다. 역시 여러 개의 decoder layer로 구성됩니다.
3. **Self-Attention + FeedForward**: 각 layer 내부에서 핵심적인 역할을 하는 두 가지 연산입니다.

## 3. Self-Attention이란?

이제 핵심인 **Self-Attention**을 설명드리겠습니다.

예를 들어, 문장 “The animal didn’t cross the street because it was too tired”가 있을 때,

“it”이 가리키는 대상을 이해하기 위해선 문맥 전체를 고려해야 합니다.

Self-Attention은 **입력의 각 단어가 다른 단어들과 얼마나 중요한 관계에 있는지를 계산**합니다.

* 각 단어는 세 가지 벡터로 변환됩니다:

**Query (Q)**, **Key (K)**, **Value (V)**

* Attention Score는 Q와 K의 내적을 통해 구하고,

소프트맥스(softmax)를 통해 가중치를 계산합니다.

* 이 가중치를 V에 곱하여 최종 출력을 만듭니다.

이 과정을 통해 각 단어는 **문장 내 모든 단어와의 관계를 고려한 새로운 벡터 표현**을 얻습니다.

## 4. Multi-Head Attention

Self-Attention을 한 번만 계산하는 게 아니라, **여러 개의 “Head”로 병렬적으로 수행**합니다.

이것이 바로 **Multi-Head Attention**입니다.

각 head는 입력을 다른 방식으로 바라보므로,

결과적으로 모델은 **여러 관점에서 문맥을 파악**할 수 있게 됩니다.

## 5. Position Encoding

Transformer는 순서를 따로 고려하지 않기 때문에, **위치 정보를 직접 추가**해야 합니다.

이를 위해 각 단어에 **Position Encoding**을 더합니다.

이는 사인/코사인 함수 기반이거나, 학습 가능한 형태로 구성됩니다.

## 6. 전체적인 Layer 구성

* **Encoder Layer**:
  + Multi-Head Self-Attention
  + Add & Norm
  + Feed-Forward Network (FFN)
  + Add & Norm
* **Decoder Layer**:
  + Masked Multi-Head Self-Attention
  + Multi-Head Attention with Encoder Output
  + FFN
  + Add & Norm

이렇게 각각 N번씩 반복됩니다. 보통 N=6.

## 7. 장점 요약

* ✅ **병렬처리 가능** → RNN보다 훨씬 빠름
* ✅ **긴 문맥 처리 가능** → 장기 의존성 문제 해결
* ✅ **해석력 뛰어남** → Attention score로 이유 추론 가능
* ✅ **확장성 우수** → BERT, GPT, T5, T5 등 다양한 파생 모델의 기반

## 8. 응용

Transformer는 이후 수많은 파생 모델의 기반이 되었습니다:

* **BERT**: 양방향 Transformer 인코더
* **GPT**: Decoder 기반의 언어 생성 모델
* **T5, BART**: 인코더-디코더 기반의 다양한 태스크 처리

또한 **음성, 이미지, 시계열(Time Series) 예측 분야**에도 적용되고 있습니다.

대표적인 예가 바로 **Temporal Fusion Transformer (TFT)** 입니다.

## 마무리

Transformer는 **자연어 처리뿐 아니라 다양한 딥러닝 분야의 기반**이 되는 핵심 구조입니다.

Attention is All You Need, 이 말이 얼마나 정확한지 지금도 계속 증명되고 있습니다.

감사합니다.

필요하시면 PPT 슬라이드 요약본이나 시각 자료도 제작해 드릴 수 있습니다.