TFX MLOps and Model Deployment

Production ML pipelines menggunakan TensorFlow Extended TFX. End-to-end workflow dari data ingestion hingga REST API serving.[1]

TFX Data Pipeline Components

Complete TFX Pipeline:

```
def data_pipeline():
    # 1. ExampleGen (data ingestion)
    example_gen = CsvExampleGen(input_base_uri='data/')

    # 2. StatisticsGen
    statistics_gen = StatisticsGen(example_gen.outputs['examples'])

    # 3. SchemaGen
    schema_gen = SchemaGen(statistics_gen.outputs['statistics'])
```

```
    # 4. ExampleValidator
    example_validator = ExampleValidator(
        statistics_gen.outputs['statistics'],
        schema_gen.outputs['schema']
    )

    # 5. Transform (feature engineering)
    transform = Transform(
        examples=example_gen.outputs['examples'],
        schema=schema_gen.outputs['schema']
    )

    return example_gen, statistics_gen, schema_gen, example_validator, transform
```

Feature Engineering dengan Transform:

```
def preprocessing_fn(inputs):
    # TensorFlow Transform operations
    x = tft.scale_to_z_score(inputs['features'])
    y = inputs['labels']
    return {'x_scaled': x}, y
```

TFX Trainer API Integration

Keras Model dalam TFX:

```
def model_builder():
    model = create_keras_model()
    model.compile(optimizer='adam', loss='mse', metrics=['mae'])
    return model

trainer = Trainer(
    module_file='trainer.py',  # Contains model_builder()
    examples=transform.outputs['transformed_examples'],
    transform_graph=transform.outputs['transform_graph']
)
```

SignatureDefs untuk Serving:

```
@tf.function(input_signature=[
    tf.TensorSpec(shape=[None], dtype=tf.string, name='input')
])
def serving_fn(input):
    # Preprocessing + model inference + postprocessing
    features = parse_features(input)
    predictions = model(features)
    return {'output': predictions}
```

Docker + TensorFlow Serving Deployment

Dockerfile:

```
FROM tensorflow/serving
COPY ./models /models
COPY ./signature.json /models/signature.json
ENV MODEL_NAME=my_model
```

REST API Serving:

```
docker run -p 8501:8501 \
  --mount type=bind,source=./models,target=/models \
  -e MODEL_NAME=my_model -t tensorflow/serving
```

Python Client:

```
import requests
import json
import tensorflow as tf

def predict_json(data):
    response = requests.post(
        'http://localhost:8501/v1/models/my_model:predict',
        data=json.dumps({"instances": data}),
        headers={'content-type': 'application/json'}
    )
    return response.json()['predictions']

# Usage
predictions = predict_json([{"feature1": 1.0, "feature2": 2.0}])
```

Model Resolution dan Versioning

Multi-model serving dengan model versioning:

```
models/
  my_model/
    1/
      saved_model.pb
    2/
      saved_model.pb  # Latest version
```

Kesimpulan

TFX menyediakan standardized, scalable ML pipelines dari data ingestion hingga production serving dengan Docker orchestration.[1]