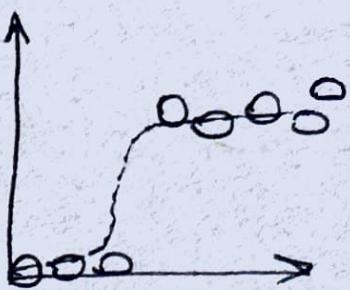


Regression
↓

Linear Regression



"Sürekli" sonuc tahmin etmek
↓
bağımsız değişken



Logistic Regression
— Classification —

↓
ikili sonuc tahmin etmek



bağımlı değişken

SVM → Support vector machine

~~Sınıflandırma
Regression~~ yöntemi. "hyperplane" bulmaya gelir

Regresyondan farklı olarak sınıfları ayırmak.

KNN → K-Nearest Neighbors

Classification + regression yöntemi

Tekin konularına göre sınıfları ayırmak

Veri Tipi:

Continuous Data - Regression:

Linear Regression
Decision Tree regressor,
Random forest regressor

Categorical Data - Classification: Logistic Regr.

SVM
Naive Bayes

Problem Tipi:

Classification → SVM, Logistic Regression,
KNN, Decision Tree

Regression → Linear Regression, SVR,
Random forest regressor

Clustering → k-means, DBSCAN

Dimensionality Reduction → PCA, LDA

Dataset Size:

Small dataset : KNN, Decision tree

Large dataset : Random forest, Gradient
Boosting, Neural Network

Number of Features:

High Number of Features: PCA, Neural network

Low number of features: KNN, Logistic regression

ALGORİTMALARI

NASIL ANIRT

EDEBİLİRİM

- cheat sheet -

problem, anketler kelime

muhtemel cevap

classification problem → logistic regre., SVM, random forest

predict continuous value → Linear regn., SVR, decision tree, regressor

Grouping, Segmentation → K-means, Hierarchical clustering

Dimensionality reduction → PCA
feature reduction

Text-data / spam detection → Naive Bayes, logistic regn

Image data → CNN

time series, sequential data → RNN, LSTM

Decision Tree basit ama "overfit" olabilir
Bu durumda Random forest ile birleştirilebilir

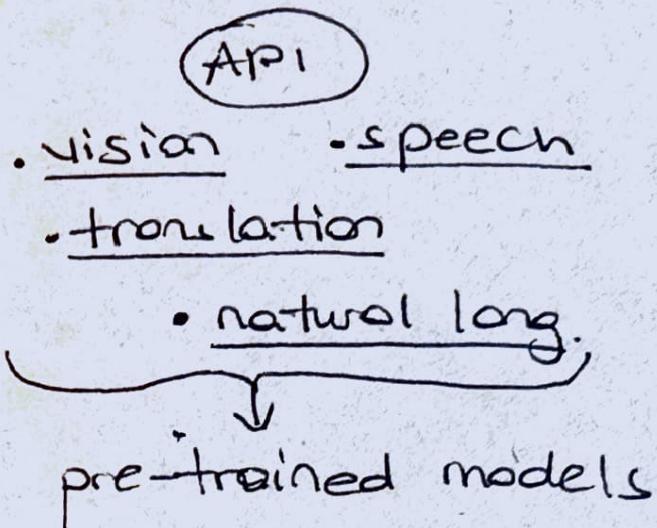
Neural network non-linear olan problemleri
gözelbilir.

ML modelinin performansını iyileştirmek için
veriye "yeni özellikler" ekleyerek, mevcut
özellikleri nüştürme.

Features arasında korelasyon bulmak için

- Pearson Coefficient → 2 Numerical features arasında
- Chi-squared test → 2 categorical features arasında
- Mutual Information → Between categorical
Information Gain → categorical to categorical
- T-Test / ANOVA → Between multi-category
and numerical feature
- PCA

Low-Code ML Solutions - Bölüm 2



Auto ML - fast

Bigquery ML

Tensorflow - most control

Build your own model

↓

developers, data analysts, ML engineers

BigQuery

- petabyte
- serverless
- flexible pricing model
- geospatial data-types / function
- BI, AI

BigQuery, iki temel
seruvisin, tek hali

- fast SQL query engine
- manage storage for datasets

Cloud Storage Google Drive

Dataflow DataPrep

Bigtable ---

→ Bigquery managed storage → Bigquery query engine

Bölüm-3

Dataproc : Bigdata işleye attiyopisini kuma, Apache Hadoop, Spark, Hive gibi bigdata erişimini hızlı şekilde cloud ortamına kuma.

Hadoop ile aynı makul deneyim yetirimi gerçekleştirir. Dataproc ile beraber kullanıldığı kodda ödersin.

Veri real-time olursa? Burada devrey dataflow girmek.

Anahtar Keline / Problem

Önerilen servis

- Streaming data, real-time processing → Dataflow pubsub
- batch processing → Dataproc
Apache spark
- Data warehouse → BigQuery
SQL Queries
- Model training, model deployment → Vertex AI
- Blob storage, data backup → Google Cloud storage
- jupyter notebooks, python development → notebooks ^{AI Platform}
- Data visualization → Looker
BI reports

Imbalanced Data

- Modelin daha büyük (çoğuluk) sınıfı öylek yıldırması, sınırlı sınıfın doğru tahmin edilememesi, bias olması

Nasıl Çözürlü?

- Sınıfları eşitlemek / püntellemek
- Doğru evaluation değerini kullanmak

Teknikler - Cheat Sheet

- ① **Up-Sampling** → Azınlık sınıfını artırmak
→
- azınlık olan veri içermiyorsa
Örnek: SMOTE (Synthetic minority oversampling technique)
- ② **Down-Sampling** → Coğuluk sınıfının örnekini azaltmak
←
- coğuluk çok veriye sahipse
Örnek: random undersampling
- ③ **Class Weight Adjustment** → Azınlık sınıfı daha fazla "ağırlık" verme
 $\text{class_weight} = \text{'balanced'}$, Logistic Regression sum,
- ④ **Anomaly Detection** → Çok deprest bir döplüm varsa, olağan olmayan kabul etmek
- ⑤ **Ensemble Methods** → Birden fazla modelin tahminlerini birleştirme, model perf. artırmak
Örnek: Random Forest, Balanced Bagging Classifier

Vertex AI Workbench

Analyze

- BigQuery
- Spark

Train

- Vertex AI Training (Automy)

automasyon

Deploy

- Vertex Prediction

Vertex Pipelines
[veri hazırlığı, model eğitimi]

Vertex AI
Workbench

Notebooks

multi-kernel in UI

Sınıflandırma Metrikleri - Cheat Sheet

Balanced Data → Accuracy

False positives are critical → Precision

False negatives are critical → Recall

Imbalanced Data → F1 Score, ROC-AUC

Note

Parameters

weights,
bias etc..

model eğitimi sırasında
değişebilir.

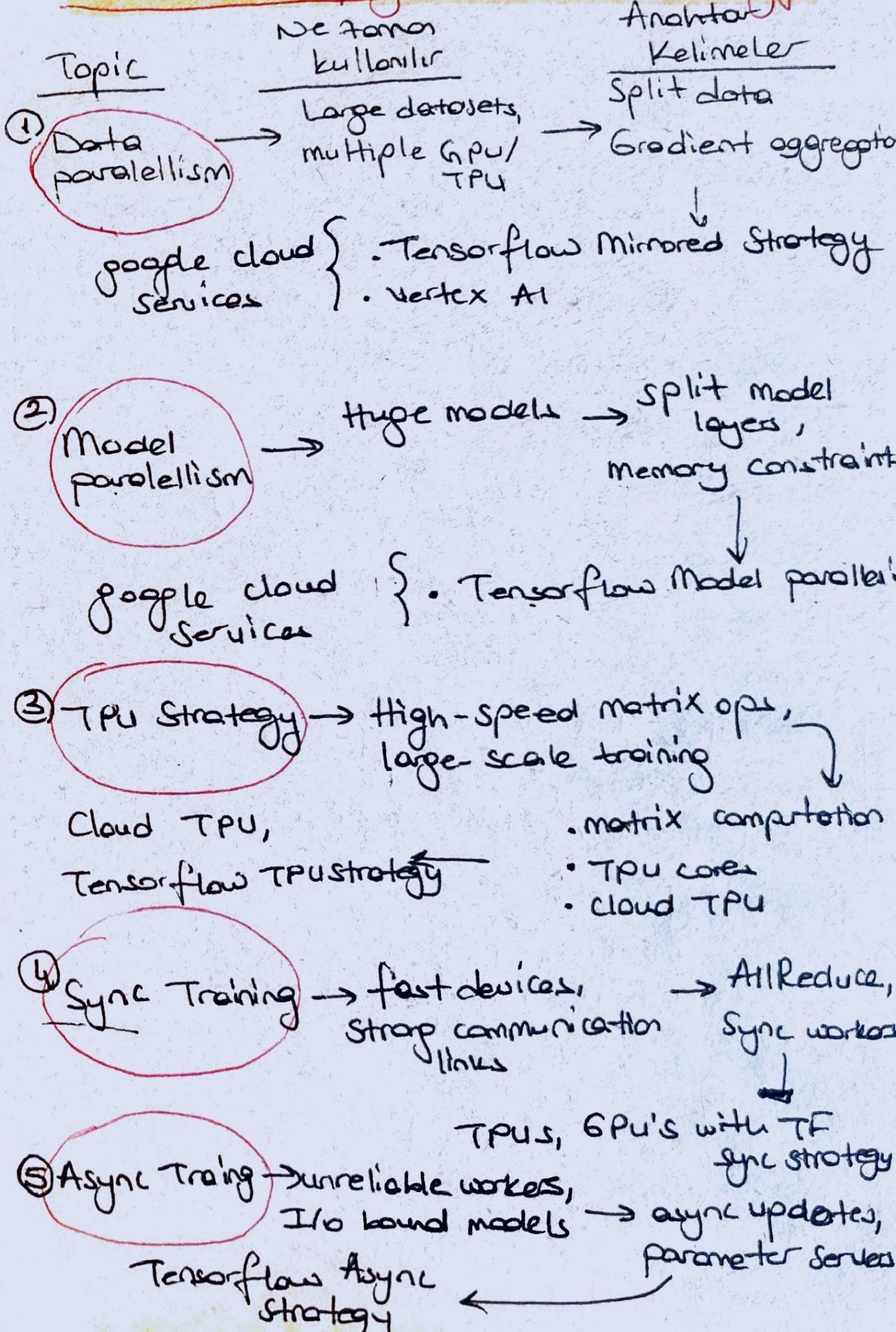
ML Model

Hyperparameters

learningrate; batch-size,

Eğitim öncesi ayarlanır.

Parallel Developing ve TPU Strategy



Data parallelism:

- split dataset into chunks
- process them across multiple devices
- senkron yada asenkron olarak Model paramet.
update
- Best for large datasets and simpler models

Model Parallelism:

- modeli cihazlar arası böler.
- model cihaz memory'sine göre böylükler kullanır

Sync , Async Parameter Server:

- tüm worker cihazlar eş zamanlı çalışın
- modeller "tutucu" olmaliysa gerekli. → örn: hassas finansal tahminler
- tüm cihazlar yüksek performanslı ve güvenliyse.

disavantage:

- eğer tek 1 worker yarasa, sistem yavaş, sistem onu beklemek zorunda kalır.
- senkronizasyon maliyeti

Tensorflow Distribution Strategy

- ① mirrored strategy = birden fazla GPU,
tek makine,
basit kurulum
- ② multi-worker mirrored = birden fazla
strategy makine
- ③ TPU Strategy = TPU özel strateji
TPU'larda hızlı eğitim
- ④ Parameter Server = model ağırlıkları
Strategy parameter server adında
node içinde saklanır.
 - modelleri ~~düzenli~~ bir ortamda
eğitmek için.

Application seviyede,

tf.distribute.Strategy

End-to-end ML Pipelines

① Vertex AI Pipelines:

- orchestrates ML workflows in a serverless manner
- tracks metadata for reproducibility

② Tensorflow Extended (TFX)

- preprocessing, training, evaluating ML model in standartized componentler
- Example Gen, Transform, Trainer, Pusher componentleri include edilmis.

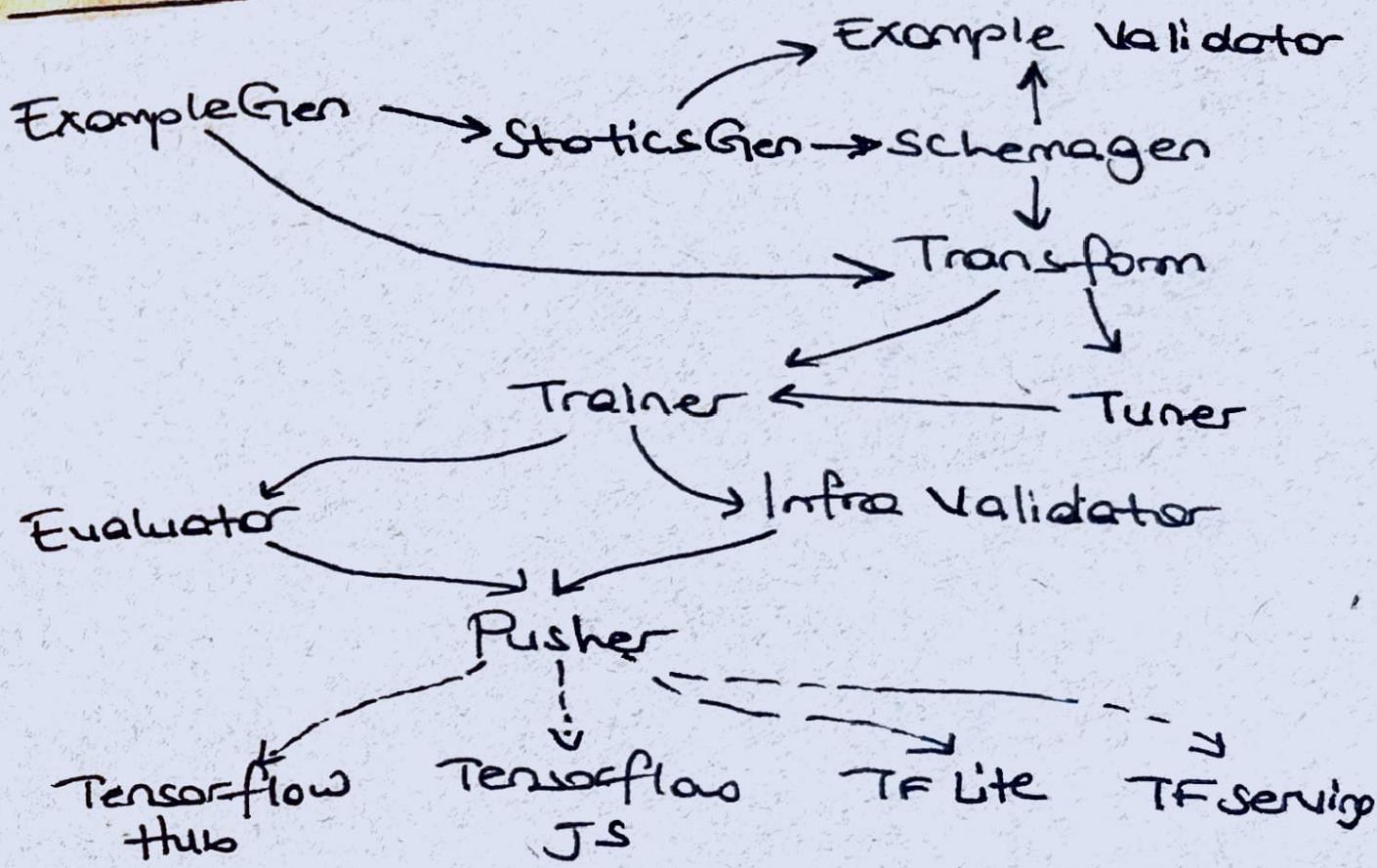
③ Kubeflow Pipelines:

- multi-framework support in flexible
- Kubernetes env. ile entegre.

MLOps Fundamentals

- Devops + ML → MLOps
- Automation + standardization for ML workflow
- "Cloud build for CI/CD pipelines on Google Cloud"
- Continuous integration (CI) → model and data validation
- Continuous delivery (CD) → deploying retrained models.

Tensorflow Extended (TFX)



Örnek Aksı:

- Example Gen verileri split eder ve alır.
- StatisticsGen data dağılımını distribute eder ve problemleri flag eder.
- SchemaGen statistics için şema oluşturur.
- ExampleValidator verilerin şemaya uygunluğunu sağlar ve anormallikleri işaretler
- Transform preprocessing steplerini yapar.
- Trainer, preprocess edilen verileri eğitime sokar.
- Evaluator model kalitesini kontrol eder.
- Infra Validator, modelin deployment uyumluluğunu sağlar.
- Pusher production için model deploy eder

Automating and Scaling Pipelines

① Scaling tools :

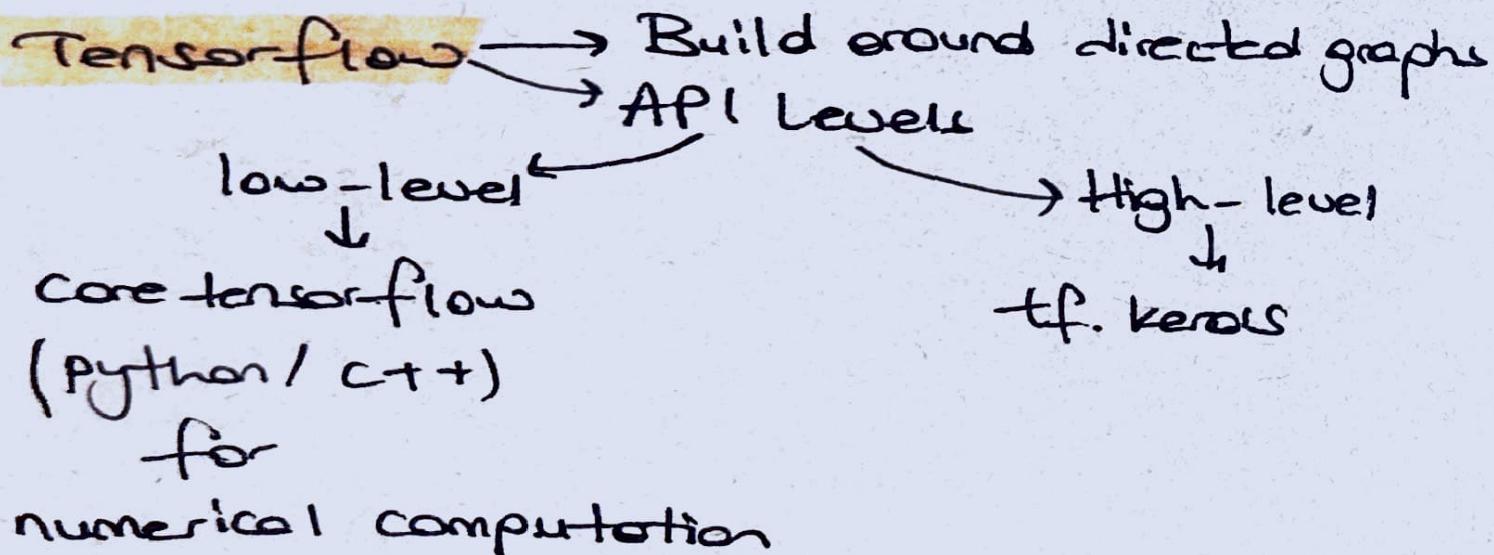
- Dataflow for distributed data pre-proc.
- AI Platform for scalable training and serving

② Orchestration:

- Vertex AI Pipelines for conditional execution
loops,
triggers

③ Integration with CI / CD

- Cloud Build, YAML based definitions for pipelines



Hyperparameter Tuning

- Learning Rate = her iterasyonda model ağırlıklarını ne kadar güncellemek.

↳ Learning rate çok düşükse; (0,00001) örn.

- Model yavaş yakinsar
- yakinsama çok yavaşça artırmak gerekir.
- yeterli epoch sayısı olmadan yakinsaymaz.

↳ Learning rate çok yüksekse; (örn: 0,5)

- model parametreleri hedefe çok büyük adımlarla ilerler.
- epochlar arası kayıp çok artar.
- kayıp (loss) eğrisi sürekli yükselsin.

Model kaybı (loss) minimize etmek için paramet gradyon iniği ile günceller. Learning rate bunu kontrol eder.

- Batch -Size = her iterasyonda kütüphane sample gradient'e girilek.

↳ too large olursa;

- Stolde updates yapar ama çok yavaş ilerler her iterasyon
- too much memory tüketir.

Number of Epochs

↳ Az olursa;

- Underfit yapsınır, yeterince öğrenmez.

↳ Çok olursa;

- Overfits olur. (early stopping kullanabilir)

Hidden Layers

↳ Az olursa;

model complex patterns'i yakalayamaz.
underfit olur.

↳ Çok olursa;

model overfit olur, noise öğrenir.

Optimizers

- Stochastic Gradient Descent (SGD)

• steepest descent using mini-batches

• learning rate ve momentum kritik hiperparam.

- Adagrad

• geçmiş gradientler ile her parametrenin
learning rate'ı orası olarak dinamik updatelər.

- Adam (Adaptive Moment Estimation)

• momentum ve learning rate combine edili.

Hyperparameter Tuning Techniques

- **Grid Search:** Belli hiperparametre değerleri aralığında kapsamlı arama.
- **Random Search:** hiperparametre kombinasyonları rastgele ömekler. Grid searchten hızlı ama daha az kapsamlı.
- **Bayesian Optimization:** requiring fewer iteration to find optimal settings.
 - uses past evaluations to guide the search

Loss Functions

- **Binary crossentropy** → 0-1 labels
- **Categorical crossentropy** → multi-class problems with one-hot encoded labels.
- **Sparse Categorical Crossentropy** → multi-class problems with integer labels (one-hot encoder değilse)

Regression Losses

- **Mean absolute error:**
 $| \text{tahminler} - \text{hedefler} |$

- **Mean Squared error:**
 $(\text{tahminler} - \text{hedefler})^2$

Tips and Tricks *

- ① Small datasets → SGD
- ② Large datasets → Adam for adaptive learning
- ③ Underfitting vs., → more layers
→ model complexity
→ larger learning-rate
- ④ Overfitting → consider dropout
→ L2 regularization
- ⑤ Jupyter-based development → Vertex AI workbench
- ⑥ BigQuery → large-scale data manipulation,
data preparation
- ⑦ Cloud Dataflow → ETL processleri için
extract transform load
data preparation tool'u.
- ⑧ Cloud Dataflow → Data processing ve
feature eng. için.

① Dataproc → managed Hadoop / Spark service.

- Batch processing için "best"
- Sisteme içi iş yüklerini bulut'a taşıma.

② Dataflow → unified batch and streaming data pipelines.

- Best for dynamic data processing and ETL tasks

③ BigQuery → Datawarehouse, SQL-loaded analytics.

④ Pub/Sub → event-driven messaging service

- support streaming data ingestion for IoT and real-time applications

⑤ Vertex AI workbench →

- managed Jupyterlab env. for end-to end ML workflows

⑥ Cloud Composer →

- workflow orchestration using Apache Airflow
- teknolojileri iş akışlarını ve pipeline yönetmek.

Donanım Standardları

CPU :

- Basit, küçük veri setleri için
- modelin ilk geliştirme aşamalarında, hata ayıklama süreçlerinde.
- paralel işlem gerçekleştirmeyen / 07 farklı görevlerde

GPU :

- yüksek paralel işlem, büyük veri seti, karmaşık modeller

TPU:

- özellikle Tensorflow için
- yüksek performans
- düşük hassasiyetli, örn: 8 bit için, ölçü verimliliği sağlar.

CPU

ne zaman
gerekli?

GPU

- model karmaşıklığı ortaya, veriseti büyürdüyse, daho hatalı eğitimi gerçekleştiriyse

GPU

ne zaman
gerekli?

TPU

- TF tabanlı büyük ve karmaşık modeller üzerinde çalıştırılsa,

Explainable AI

① Model açıklayılabilir olmalı:

- basit modeller daha onaylıdır.
- modelin karar alırken hangi özelliklere ne kadar önem verdiği analiz edilmelidir.
- Vertex AI, SHAP (SHapley Additive Explanations) LIME (Local Interpretable model ex -) gibi oruçlar ile model koşulları açıklayıcı hole getirir.
→ Bu oruçlar tahminin hangi girdilere dayanarak yapıldığını gösterleştirir.

② Çıktı açıklayılabilir olmalı:

- çıktılar püskürtilebilir, örn: lsı haritalama, önemli kelimeleri veya çalışma
- metrik değer sonucu ve riskleri belirtilebilir.

③ Ürun açıklayıcı olmalıdır, arayüzde açıklandırılabilir

④ Standartlara uygunluk :

- Fairness - adalet,
- Transparency - Eşeffektlik
- Accountability - Sorumluluk

Ne forman Backbone Olur?

- ① Yerel Trafik
- ② Hatalı iş yükü dağılımı
- ③ Kaynak Metersitliği
- ④ Verilerin doğru node'lara iletilmemesi

Örn: CPU %80 üzerinde çalışıyor.

Örn: Pub/Sub mesaj kuyruğunda kırıkma oldu.

Nasıl müdahale ederim?

- ① Scaling
 - Vertical Scaling = Daha güçlü CPU/Bellek ekleme
 - Horizontal Scaling = Daha fazla instance, Cloud Run, Kubernetes pod sayısını artırma
 - Autoscaling
- ② İş yükü Dağılımı → Load Balancer kullanma
 - ↳ Geleneksel Pub/Sub kullanın
- ③ Kaynak izleme/Uyarı → "Metrics explorer"
 - ↳ CPU, bellek, ağ kullanımını izler
 - Alerting
- ④ Caching → Cloud Memory store
 - ↳ en çok erişilen veriler cache ile tutulabilir.
 - Redis
- ⑤ Veri Bölme (Sharding)
- ⑥ Kod optimizasyonu → Profiler - hangi işlemekin en çok kaynak tükettiğini analizi

Barcode Olmamasi Icm

- ① Cloud Monitoring → sistem metriklerini itle.
- ② Load Balancing, yükü dengele.
- ③ Auto-scaling - Compute Engine Autoscaler
 - Serverless uygulamalar → cloud run
- ④ Olay Tabanlı mirası kullanımları
 - pub / sub
 - Dataflow

Monitoring ve Utilization Kod Parçaları

- ① CPU Kullanımı → cpu-metric = " "
filter-expression =
- ② Uygun olusturma → threshold_value =
- Threshold - duration =
- ③ Trafigi Bölgeleri itleme → region-filter = " "
resource.labels.zone
- ④ Aktif Bağlantıları itleme → active-conne-metric =
"loadbalancing.google..."
- Load Balancer -

Real-time Services

- ① Pub/Sub → gercek zamanlı mesajlaşma,
Event-driven iletim
- ② Dataflow → Gerçek zamanlı veri işleme,
ETL (Extract, Transform, Load) gibi
Apache Beam ile uyumlu. - streaming, batch
process.
Otomatik ölçeklenir.
Low-latency - gerçek zamanlı analizler
ömrü: sosyal medya verisi analizi
- ③ BigQuery Streaming → Real-time data analytic
- ④ Cloud Spanner → Real-time data management,
Globally scalable DB
- online oyun, veri göretimi,
Bu listeyle birlikte Cloud Run,
Cloud functions

Kombinasyonlar

- ① Pub/Sub + Dataflow → Veritabanı, işleme,
analiz
- ② Cloud Monitoring → real-time sistem
performansı
- ③ AI Prediction
or
BigQuery Streaming → Anlık raporlama,
tahmin süreçleri

Kubernetes Services

- Açık kaynok, Google 'in geliştirdiği,
"konteyner" orkestrasyon servisi"
- Bir uygulama ve bağımlılıkları bir arada
tutar. - Dağıtım, Ölçekleme, Yönetim otomatiklikleri.

Kubernetes Temel İşlevi:

- ① Otomatik Ölçekleme = kullanım artarsa,
daha fazla konteyner ekler (horizontal scaling)
kaynak tüketimini optimize eder.
- ② Bir konteyner ontolanırsa hemen otomatik
yenilimi başlatır.
Uygulamalar kesintisiz kalır.
- ③ Yük Dengeleme, Load Balancing
Gelen yük konteynerlara dağılıp iletilir
- ④ Otomasyon,
Konteyner yasaklı操作 yönetimin
bağlantı, durdurma, geriçellene

Google Kubernetes Engine (GKE)

- Tamamen yönetilebilir bir servis.
- Abyopi yönetimi: Google'nın, Kubernetes'i kurmak, yönetmek ve optimize etmek için çok iyi harcamasıdır.
- Birden fazla konteyneri bir orada yönetir, alt türler
- Özellikle büyük, kompleks uygulamaları doğrudan kullanılır.
- Büyük veri, AI projelerinde doğrudan iş yüklerini cağırtır. "mikro hizmet mimarisi"
- Node'lar, cluster'lar ve doğrudan dimensiye döngü-fazla kontrol edebilirsin.
- Kaynaklar otomatik olarak kullanılabilir.

GKE ile Sıkça Kullanılan Servisler

1. Cloud Storage
2. Cloud Pub/Sub
3. BigQuery
4. Vertex AI
5. Cloud Monitoring (Operations Suite)

Sesliler ve Aşağıtak Kelimeler

Compute Engine → VM - IaaS

GPU, TPU desteği
Sunucu yönetimi sende.

Cloud Run → Konteyner tabanlı, Serverless (~~PaaS~~)
Küçük otomatik uygulamalar.
Abyapıyi dízünmüþürüm.

App Engine → PaaS,
kod yaz, yükle, uygulama getirsin.
Gök düşük düzey kontrol

Dataproc → Hadoop, Spark, Hive gibi bigdata
araclarını birleştirir.

- Spark MLlib gibi araclarla ML modelleri oluşturma
- Hive, SparkSQL ile big data analizi
 - petabyte veri işleme
 - Batch processing
 - gerçek zamanlı ise uygun değil

Cluster — Birden fazla sunucu ama tek sistem

- ① Sunucusu (Node) uyum içinde çalışır.
 - ② Görevler dağıtılar, iş yükü dengeLENİR.
 - ③ node çalışmazsa, hemen diğerleri devreye girer.
 - ④ sistem büyürse daha fazla node eklenir.
 - ⑤ cluster boyutundan sistemlerin hata toleransı var.
- Master Node - Cluster'i yöneten ana sunucu
↳ hangi görev, hangi node'da çalıştırılacak onu belittir.
- Worker Node - Görevli Node'lar, Çalıştırılır
- Load Balancer - Node'lar extt miktarında yük alır.
"Dengeler"

Cluster Kullanım GCP Hizmetleri

- ① Kubernetes Engine → master node, worker node konteyneri çalıştırmaK.
- ② Dataproc → Hadoop, spark, veri işleme clusterları
- ③ Bigtable → geniş ölçekte veri saklama,
→ yüksek erişim, düşük gecikme

Not - Pod → Kubernetes'de bir veya daha fazla konteynerin çalıştığı "blim"

Veri Depolama Servisleri

- ① BigTable - NoSQL DB - TB seviyesi
- Büyük ölücek, düşük gecikme (finans)
- Sıralı okuma, yazma için hızlı.
- analitik değil, işlem/erişim odaklı.
- Hiperörsiz değil, sütun bazlı veri saklar.
- ② Spanner - Cloud SQL gibi ama global ölücekli.
- Bankacılık, finans, e-ticaret
- ③ DataStore - NoSQL, Document Store
- Düşük ölücekli NoSQL uygulamaları
- firestore'un eski versiyonu

! Cheat sheet !

- ④ Analitik → BigQuery @İşlem → BigTable, Firestore,
Cloud SQL
- ⑤ SQL → Cloud-SQL @NoSQL → BigTable
Spanner
Firestore
Datastore

- ⑥ Gerçek
Zamanlı → Firestore
(Real-time) → BigTable