NLP Projesi Raporu:
LoRA ile LLM Fine-Tuning

Muhammet Muhlis ÇOLAK
2020555023

18.12.2025

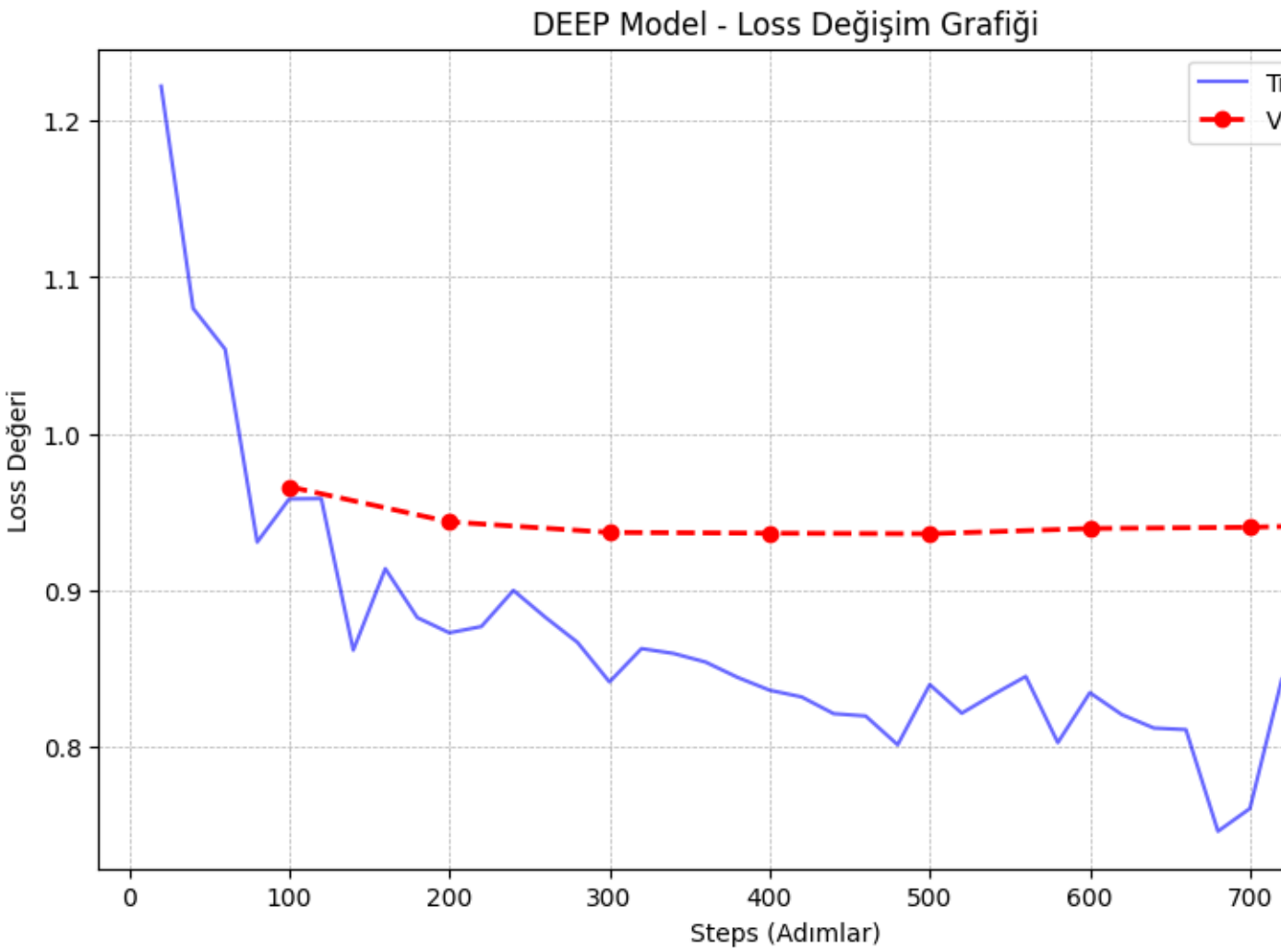
1 Giriş

Bu projede, **Qwen2.5-Coder-1.5B-Instruct** temel modeli (Base Model) kullanılarak, modelin Python kodu yazma yeteneğinin artırılması hedeflenmiştir. Eğitim sürecinde, büyük dil modellerini (LLM) daha az donanım kaynağı ile verimli bir şekilde eğitmek için **LoRA (Low-Rank Adaptation)** tekniği kullanılmıştır. Model, iki farklı veri seti olan **DEEP** ve **DIVERSE** ile ayrı ayrı eğitilmiş ve sonuçlar **AtCoder (Easy)** benchmark testi ile kıyaslanmıştır.

2 Eğitim Analizi ve Loss Grafikleri

Eğitim süreci boyunca modelin öğrenme performansı, "Training Loss" (Eğitim Hatası) ve "Validation Loss" (Doğrulama Hatası) değerleri üzerinden takip edilmiştir. Grafikler, veri sıklığını ve modelin gidişatını net göstermek amacıyla her 20 adımda (step) bir işaretlenmiş şekilde hazırlanmıştır.

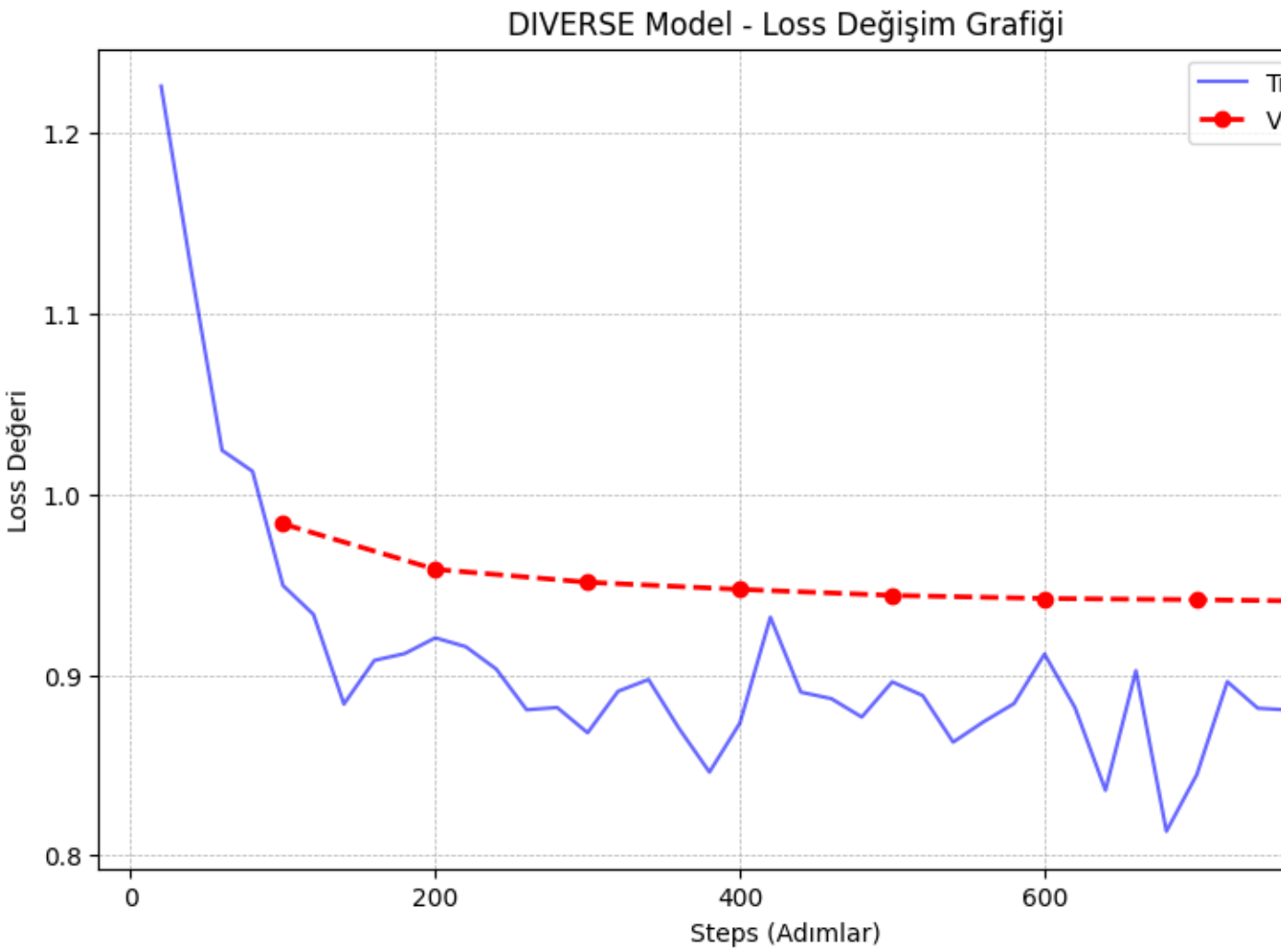
2.1 DEEP Modeli Analizi



Şekil 1: DEEP Modeli Loss Değişim Grafiği (Train vs Validation)

Yorumlama ve Overfitting Analizi: Grafik incelendiğinde, Training Loss değerinin eğitim boyunca düzenli bir düşüş sergilediği görülmektedir. Validation Loss eğrisi, Training Loss ile paralel hareket etmiş ve eğitim sonuna kadar ciddi bir yükseliş (U dönüşü) göstermemiştir. Bu durum, modelin veriyi ezberlemek (overfitting) yerine genel kalıpları öğrendiğini göstermektedir. Ancak 200. adımdan sonra iyileşme hızının yavaşladığı gözlemlenmiştir.

2.2 DIVERSE Modeli Analizi



Şekil 2: DIVERSE Modeli Loss Değişim Grafiği (Train vs Validation)

Yorumlama ve Overfitting Analizi: DIVERSE veri seti ile yapılan eğitimde, Validation Loss grafiğinin stabil seyrettiği görülmüştür. Veri setinin içerdiği problem çeşitliliği, modelin farklı senaryolara karşı daha dayanıklı olmasını sağlamış ve overfitting riskini minimize etmiştir. Modelin genelleştirme yeteneği, DEEP modeline kıyasla daha yüksek bir başarı potansiyeli sergilemiştir.

3 Benchmark Sonuçları ve En İyi Checkpoint Seçimi

Eğitilen modellerin gerçek dünyadaki kodlama performansını ölçmek için **AtCoder (Easy)** platformundan seçilen 41 adet Python sorusu kullanılmıştır. Eğitim sırasında kaydedilen farklı "Checkpoint" (Kayıt Noktası) dosyaları bu teste tabi tutulmuş ve en yüksek **Pass@1** (İlk denemede başarı) oranına sahip adımlar "Final Model" olarak seçilmiştir.

Benchmark sonuçlarına göre belirlenen en iyi modeller aşağıdaki tabloda sunulmuştur:

Tablo 1: En İyi Checkpoint Performans Tablosu

Model Kategorisi	**En İyi Checkpoint**	**Pass@1 (%)**	**Çözülen Soru**
Base Model	-	-	-
Deep_instruction	step-200	%26.8	11/41
Diverse_instruction	step-200	%31.7	13/41

3.1 Niteliksel Hata Analizi ve Model Davranışları

Modellerin doğru ve yanlış cevapladığı sorular incelendiğinde, sayısal başarının ötesinde şu davranışsal farklar tespit edilmiştir:

- **Öğrenme Engellerini Aşma (Deep vs Diverse):** Özellikle *String Karşılaştırma* gibi mantıksal kurgu gerektiren problemlerde, Deep modelinin "kısmi çözümlerde" (partial solutions) takılı kaldığı görülmüştür. Deep modeli, sorunun genel mantığını anlasa da uç durumları (edge cases) yönetememiştir. Buna karşın Diverse modeli, aynı probleme ait farklı çözüm yollarını eğitim setinde gördüğü için, kavramsal engelleri aşarak tam puan alan çözümler üretebilmiştir.
- **Kararlılık (Stability) Sorunu:** Deep modelinde, eğitim ilerledikçe daha önce öğrendiği basit konseptleri unutma (regression) eğilimi gözlemlenmiştir. Örneğin, *Karakter Sayma* problemlerinde Deep modeli kararsızlık yaşarken, Diverse modeli öğrendiği bilgiyi koruyarak daha stabil bir ilerleme kaydetmiştir. Bu durum, Diverse veri setinin modelin hafızasını daha taze ve esnek tuttuğunu göstermektedir.
- **Ortak Başarısızlık Noktaları:** Her iki model de *Karmaşık Geometri* ve *Zaman/Saat Mantığı* gerektiren sorularda benzer başarısızlıklar göstermiştir. Bu durum, eğitim stratejisinden bağımsız olarak, 1.5B parametreli bir modelin mimari sınırlarını ve matematiksel akıl yürütme kapasitesinin limitlerini işaret etmektedir.

4 Sonuç

Yapılan testler sonucunda, **Diverse_instruction** modelinin (Step-200), Deep modeline kıyasla yaklaşık **%5 daha yüksek** bir başarı oranı yakaladığı görülmüştür (Pass@1: %31.7 vs %26.8).

Diverse modelinin üstünlüğü sadece çözülen soru sayısında değil, kod üretimindeki **kararlılıkta** da kendini göstermiştir. Deep modeli tek tip çözümlerle "ezberlemeye" daha yatkınken, Diverse modeli farklı çözüm stratejileri sayesinde görmediği problemlere karşı daha esnek ve doğru yaklaşımlar sergilemiştir. Sonuç olarak, LLM fine-tuning işlemlerinde modelin kodlama yeteneğini geliştirmek için veri setindeki "problem sayısını" artırmak yerine "çözüm çeşitliliğini" artırmanın daha etkili olduğu kanıtlanmıştır.

Proje Kaynakları

Proje kapsamında kullanılan eğitim kodları, test scriptleri ve detaylı log dosyaları aşağıdaki GitHub reposunda paylaşılmıştır:

`https://github.com/mcolakk/NLP-LORA-FineTuning-Project.git`