BLM5109 1. ÖDEV

OPTİMİZASYON ALGORİTMALARINI KARŞILAŞTIRMA

Büşra Medine GÜRAL
20011038
medine.gural@std.yildiz.edu.tr

1. GİRİŞ

Bu çalışmada, Stochastic Gradient Descent (SGD), SGD with Momentum ve Adam olmak üzere üç farklı optimizasyon algoritmasının performansları karşılaştırılmıştır. Bu algoritmalar, görüntü tabanlı MNIST ve metin tabanlı Stanford Sentiment Treebank (SST) veri setleri üzerinde test edilmiştir. Her algoritma, beş farklı başlangıç noktasından başlatılarak optimize edilmiş ve sonuçlar t-SNE yöntemiyle görselleştirilmiştir.

Deneyin amacı, farklı optimizasyon algoritmalarının yakınsama hızlarını, trajektoryalarını ve genel performanslarını analiz ederek veri türüne bağlı olarak avantajlarını ve dezavantajlarını değerlendirmektir. Kullanılan modeller, her iki veri seti için de basit ve hafif bir yapıda tasarlanmıştır. Sonuçlar, algoritmaların çeşitli veri türlerindeki etkinliklerini kıyaslamaya olanak sağlamıştır.

2. KULLANILAN VERİ SETLERİ VE MODELLER

2.1 Veri Setleri

Bu çalışmada iki farklı veri seti kullanılmıştır:

a. **MNIST**:
   - **Açıklama:** El yazısı rakamların sınıflandırılmasına yönelik bir görüntü veri setidir. Her görüntü, 28x28 piksel boyutunda gri tonlamalı bir rakamdan oluşur.
   - **Sınıf Sayısı:** 10 (0'dan 9'a kadar).
   - **Veri Dağılımı:**
     - Eğitim seti: 60.000 görüntü.
     - Test seti: 10.000 görüntü.
   - **Özellik**: Görüntülerin sınıflandırılması için yaygın olarak kullanılan, basit ama etkili bir benchmark veri setidir.

b. **Stanford Sentiment Treebank (SST)**:
   - **Açıklama**: Duygu analizi için kullanılan bir metin veri setidir. Cümlelerin pozitif veya negatif duygu içerip içermediğini sınıflandırmak için tasarlanmıştır.
   - **Sınıf Sayısı**: 2 (Pozitif, Negatif).
   - **Veri Dağılımı**:
     - Eğitim seti: ~8.500 cümle.
     - Test seti: ~2.200 cümle.
   - **Özellik**: Metin tabanlı bir veri seti olup, embedding tabanlı doğal dil işleme yöntemlerini değerlendirmek için uygundur.

2.2 Modeller

Bu çalışmada, MNIST ve SST veri setleri için iki farklı model tasarlanmıştır. Görüntü tabanlı MNIST veri seti için tasarlanan model, üç tam bağlantılı katmandan oluşmaktadır. İlk katman, 28x28 boyutundaki görüntüleri 128 boyutlu bir vektöre dönüştürmekte, ikinci katman ise bu vektörü 64 boyuta indirgemektedir. Son katman, 10 farklı sınıf için softmax skorları üretmektedir. Modelde, her tam bağlantılı katmandan sonra ReLU aktivasyon fonksiyonu kullanılmış ve giriş verileri düzleştirilerek işlenmiştir.

Metin tabanlı SST veri seti için geliştirilen model ise bir embedding katmanı ve bir tam bağlantılı katmandan oluşmaktadır. Embedding katmanı, kelimeleri belirli bir boyutta temsil etmekte ve metindeki tüm kelimelerin embedding değerlerinin ortalaması alınarak bir vektör elde edilmektedir. Bu vektör, tam bağlantılı katman aracılığıyla işlenerek pozitif veya negatif sınıflardan birine ait softmax skorları üretilmektedir. Model, metin verisi üzerinde hafif ve hızlı bir yapı sunarak optimizasyon algoritmalarını karşılaştırmak için uygundur.

Model	Kullanılan Veri Seti	Girdi Boyutu	Çıkış Boyutu	Katman Sayısı	Aktivasyonlar
MNISTModel	MNIST	784 (28x28)	10	3	ReLU
SSTModel	SST	Token Dizi	2	2	None (Embedding'den sonra pooling kullanılır)

3. KULLANILAN OPTİMİZASYON ALGORİTMALARI

Bu çalışmada, optimizasyon sürecinde üç farklı algoritma kullanılmıştır: Stochastic Gradient Descent (SGD), SGD with Momentum ve Adam. Bu algoritmalar, modellerin parametrelerini güncelleyerek kayıp fonksiyonunu minimize etmek için farklı stratejiler benimsemektedir. SGD, her bir mini-batch üzerinde kayıp fonksiyonunun gradyanını hesaplayarak model parametrelerini günceller. Basit ve etkili bir yöntem olmasına rağmen, yüksek osilasyon nedeniyle kararlı bir şekilde yakınsama süreci zaman alabilmektedir. Bu sorunu aşmak için kullanılan SGD with Momentum algoritması, optimizasyon sürecine hız kazandırmak ve osilasyonları azaltmak amacıyla önceki gradyan güncellemelerinin bir kısmını güncel güncellemeye ekler. Momentum, yüksek eğimli bölgelerde daha kararlı bir yakınsama sağlayarak algoritmanın performansını artırır.

Adam, adaptif bir optimizasyon algoritmasıdır ve hem momentum hem de RMSProp yöntemlerini birleştirerek çalışır. Her parametre için ayrı ayrı öğrenme oranları hesaplayan Adam, özellikle karmaşık veri setlerinde ve yüksek boyutlu parametre uzaylarında etkili bir optimizasyon sunmaktadır. Bu algoritma, SGD'ye kıyasla daha hızlı yakınsama ve daha kararlı bir performans sergilemektedir. Çalışmada, her üç algoritma da MNIST ve SST veri setlerinde farklı başlangıç noktalarından başlatılmış, yakınsama hızları ve trajektoryaları t-SNE yöntemiyle görselleştirilerek analiz edilmiştir. Algoritmaların performansları, kayıp fonksiyonundaki değişimler ve trajektoryaların genel yapıları üzerinden karşılaştırılmıştır.

4. DENEYSEL PARAMETRELER

Bu çalışmada, farklı optimizasyon algoritmalarını değerlendirmek için belirli deneysel parametreler kullanılmıştır. Her iki veri seti için de aynı temel yapı korunmuş, ancak veri türüne uygun modeller ve işlem adımları tercih edilmiştir.

a. Model eğitimi, her algoritma için 20 epoch boyunca gerçekleştirilmiştir. Bu, algoritmaların yakınsama hızını ve uzun vadeli davranışlarını gözlemlemek için yeterli bir süre sağlamıştır.
    - num_epochs = 20
b. Her algoritma, beş farklı başlangıç ağırlığından başlatılarak eğitilmiştir. Bu, algoritmaların farklı başlangıç noktalarında ne kadar kararlı olduğunu ve performanslarının nasıl değiştiğini analiz etmek için yapılmıştır.
    - num_initialization = 5

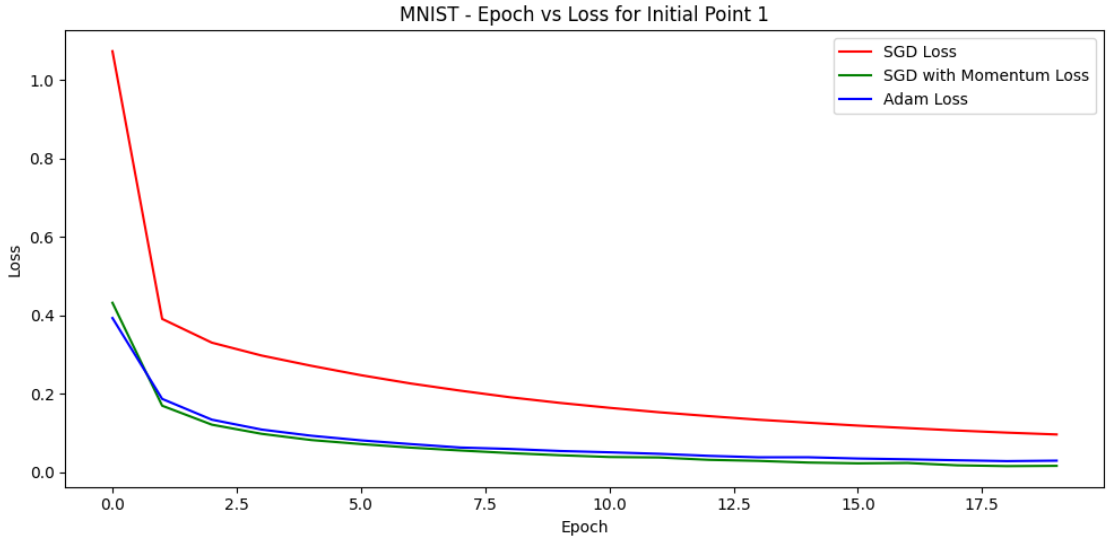
c. Eğitim sırasında kullanılan batch boyutu 128 olarak belirlenmiştir. Bu boyut hem optimizasyon sürecini hızlandırmış hem de bellek kullanımını dengelemiştir.
    - batch_size = 128
d. Modelin performansını değerlendirmek için çapraz entropi kayıp fonksiyonu (CrossEntropyLoss) kullanılmıştır. Bu kayıp fonksiyonu, sınıflandırma problemlerinde yaygın olarak kullanılan bir ölçüttür ve modelin tahminlerinin doğru sınıflarla ne kadar uyumlu olduğunu ölçer.
    - criterion = nn.CrossEntropyLoss()
e. Deneylerde üç farklı optimizasyon algoritması kullanılmıştır. Bu algoritmalar, get_optimizers fonksiyonu ile tanımlanmış ve her model için farklı başlangıç noktalarında test edilmiştir:

    - **Stochastic Gradient Descent (SGD)**:
        - Sabit bir öğrenme oranı olan lr=0.01 ile kullanılmıştır.
        - Model parametrelerini kayıp fonksiyonunun gradyanına göre günceller.
    - **SGD with Momentum**:
        - SGD algoritmasına momentum (momentum=0.9) eklenerek osilasyonların azaltılması ve daha hızlı yakınsama sağlanmıştır.
        - Öğrenme oranı lr=0.01 olarak belirlenmiştir.
    - **Adam**:
        - Adaptif öğrenme oranı kullanan Adam algoritmasında, başlangıç öğrenme oranı lr=0.001 olarak seçilmiştir. Bu algoritma hem birinci moment hem de ikinci moment (gradyan karelerinin ortalaması) tahminlerini birleştirerek daha etkili optimizasyon sağlar.

5. SONUÇLAR VE ANALİZLER

5.1 MNIST Veri Seti

MNIST veri seti üzerinde yapılan deneyde, her bir noktadan başlayan algoritmaların epoch-loss ve time-loss grafikleri aşağıdaki gibi verilmiştir.



Bu grafikte, üç farklı optimizasyon algoritmasının belirlenen ilk başlangıç noktasından başlayarak her epoch sonunda kayıp değerindeki değişim gösterilmektedir. Grafiğe ait detaylı değerler tablodaki gibidir.

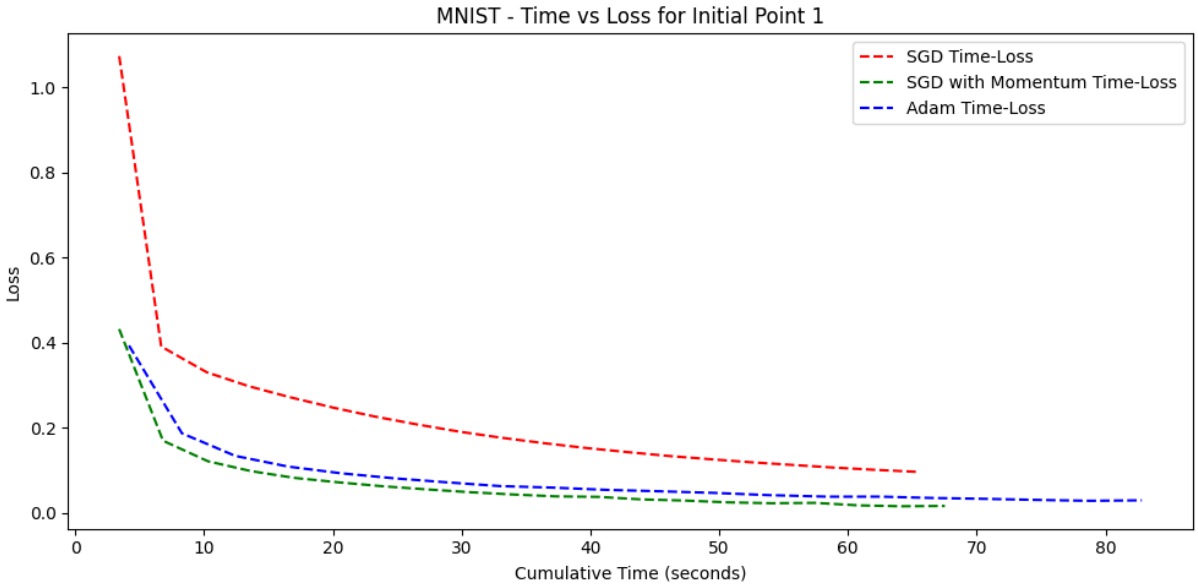
Optimizer	Epoch	Loss
SGD	0	1,0735
SGD	1	0,3909
SGD	2	0,3303
SGD	3	0,2975
SGD	4	0,2714
SGD	5	0,2474
SGD	6	0,2262
SGD	7	0,208
SGD	8	0,1911
SGD	9	0,1768
SGD	10	0,1642
SGD	11	0,1529
SGD	12	0,1431
SGD	13	0,1338
SGD	14	0,1262
SGD	15	0,1188
SGD	16	0,1125
SGD	17	0,1063
SGD	18	0,1009
SGD	19	0,0962
SGD with Momentum	0	0,432
SGD with Momentum	1	0,1696
SGD with Momentum	2	0,1212
SGD with Momentum	3	0,0979
SGD with Momentum	4	0,082
SGD with Momentum	5	0,0718
SGD with Momentum	6	0,0627
SGD with Momentum	7	0,0554
SGD with Momentum	8	0,0487
SGD with Momentum	9	0,0433
SGD with Momentum	10	0,0387
SGD with Momentum	11	0,0375
SGD with Momentum	12	0,0315
SGD with Momentum	13	0,0288
SGD with Momentum	14	0,0246
SGD with Momentum	15	0,0226
SGD with Momentum	16	0,0234
SGD with Momentum	17	0,0176
SGD with Momentum	18	0,0155
SGD with Momentum	19	0,0162

Adam	0	0,3931
Adam	1	0,1869
Adam	2	0,1339
Adam	3	0,1087
Adam	4	0,093
Adam	5	0,0812
Adam	6	0,0717
Adam	7	0,0629
Adam	8	0,0593
Adam	9	0,0541
Adam	10	0,0507
Adam	11	0,0469
Adam	12	0,0417
Adam	13	0,038
Adam	14	0,0381
Adam	15	0,0349
Adam	16	0,0329
Adam	17	0,0304
Adam	18	0,0283
Adam	19	0,0294

Kırmızı çizgi, SGD algoritmasını temsil etmektedir. İlk epoch'ta kayıp değeri hızlı bir şekilde azalsa da (yaklaşık 1.07'den 0.39'a), sonraki epoch'larda azalma oranı belirgin bir şekilde yavaşlamaktadır. SGD'nin doğası gereği, osilasyonlara açık olması ve sabit öğrenme oranı kullanması nedeniyle, kayıp fonksiyonunda daha yavaş bir yakınsama gözlemlenmektedir. Özellikle epoch sayısı arttıkça azalma hızı giderek düşmektedir.

Yeşil çizgi, momentum eklenmiş SGD algoritmasını göstermektedir. İlk birkaç epoch boyunca kayıp değerinde hızlı bir azalma sağlanmış, 4. epoch'tan sonra ise yakınsama daha stabil bir hale gelmiştir. Momentum terimi, SGD'nin osilasyonlarını azaltarak daha kararlı bir yakınsama sağlamaktadır. Kayıp değeri, SGD'ye kıyasla daha düşük değerlere ulaşmıştır ve yaklaşık 0.016 civarında stabilize olmuştur.

Mavi çizgi, adaptif öğrenme oranı kullanan Adam algoritmasını göstermektedir. Adam, özellikle ilk birkaç epoch boyunca SGD ve SGD with Momentum'a kıyasla çok hızlı bir yakınsama sağlamaktadır. Yakınsama hızı ilk birkaç epoch'tan sonra yavaşlasa da kayıp değeri en düşük seviyelere ulaşmıştır (yaklaşık 0.03). Adaptif öğrenme oranı sayesinde Adam, özellikle optimizasyonun başlarında yüksek bir performans sergilemiştir.



Bu grafikte, üç farklı optimizasyon algoritmasının belirlenen başlangıç noktasından itibaren zamanla kayıp değerindeki değişim gösterilmektedir. Grafiğe ait detaylı değerler tablodaki gibidir.

Optimizer	Time	Loss
SGD	3,41	1,0735
SGD	6,67	0,3909
SGD	10,21	0,3303
SGD	13,48	0,2975
SGD	16,72	0,2714
SGD	20,02	0,2474
SGD	23,27	0,2262
SGD	26,51	0,208
SGD	29,76	0,1911
SGD	33,02	0,1768
SGD	36,28	0,1642
SGD	39,53	0,1529
SGD	42,78	0,1431
SGD	46,01	0,1338
SGD	49,24	0,1262
SGD	52,5	0,1188
SGD	55,73	0,1125
SGD	58,98	0,1063
SGD	62,23	0,1009
SGD	65,48	0,0962
SGD with Momentum	3,41	0,432
SGD with Momentum	6,81	0,1696
SGD with Momentum	10,31	0,1212
SGD with Momentum	13,67	0,0979
SGD with Momentum	17,06	0,082

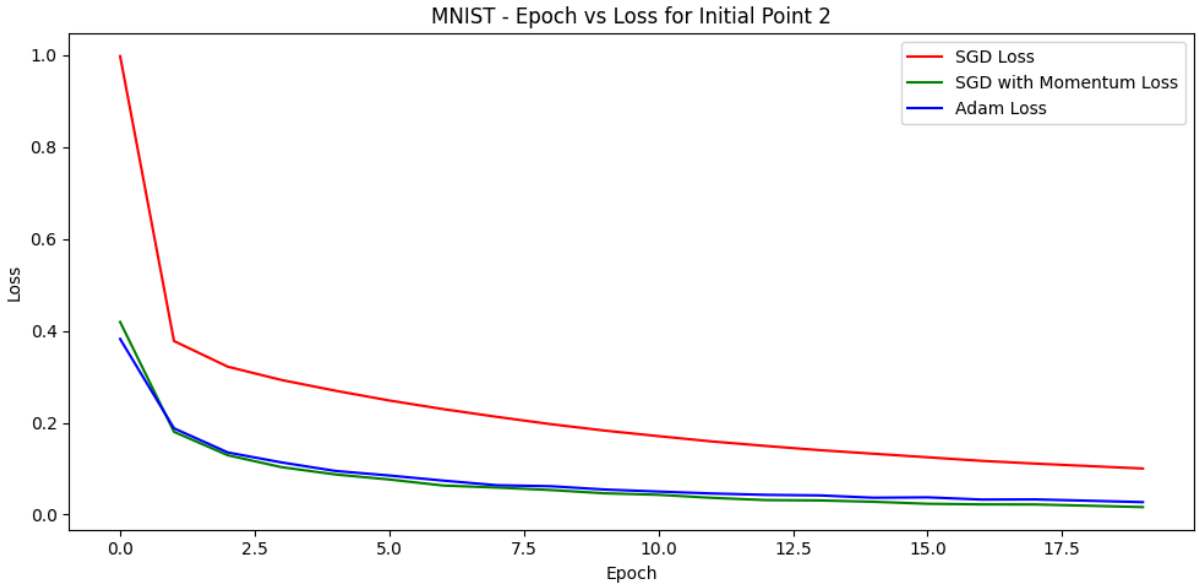
SGD with Momentum	20,42	0,0718
SGD with Momentum	23,8	0,0627
SGD with Momentum	27,16	0,0554
SGD with Momentum	30,53	0,0487
SGD with Momentum	33,9	0,0433
SGD with Momentum	37,27	0,0387
SGD with Momentum	40,63	0,0375
SGD with Momentum	43,99	0,0315
SGD with Momentum	47,35	0,0288
SGD with Momentum	50,72	0,0246
SGD with Momentum	54,08	0,0226
SGD with Momentum	57,45	0,0234
SGD with Momentum	60,82	0,0176
SGD with Momentum	64,18	0,0155
SGD with Momentum	67,53	0,0162
Adam	4,17	0,3931
Adam	8,28	0,1869
Adam	12,42	0,1339
Adam	16,53	0,1087
Adam	20,64	0,093
Adam	24,75	0,0812
Adam	28,86	0,0717
Adam	32,98	0,0629
Adam	37,09	0,0593
Adam	41,19	0,0541
Adam	45,29	0,0507
Adam	49,6	0,0469
Adam	53,81	0,0417
Adam	58,45	0,038
Adam	62,56	0,0381
Adam	66,63	0,0349
Adam	70,64	0,0329
Adam	74,71	0,0304
Adam	78,77	0,0283
Adam	82,83	0,0294

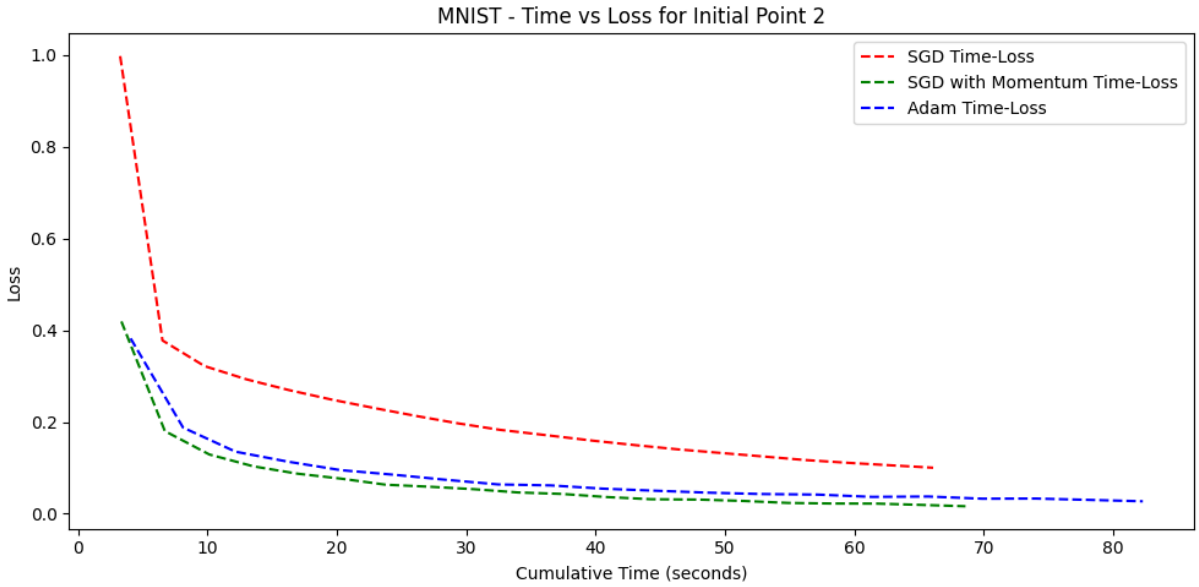
Kırmızı kesikli çizgi, SGD algoritmasını temsil etmektedir. Kayıp değeri, yaklaşık 60 saniyeye kadar oldukça yavaş bir şekilde azalmaktadır. SGD'nin sabit bir öğrenme oranı ile çalışması, zaman açısından daha düşük bir performans sergilemesine neden olmaktadır. İlk epoch'larda belirgin bir kayıp azalımı sağlasa da toplam süre boyunca kayıp değeri diğer algoritmalara kıyasla daha yüksek seviyelerde kalmıştır.

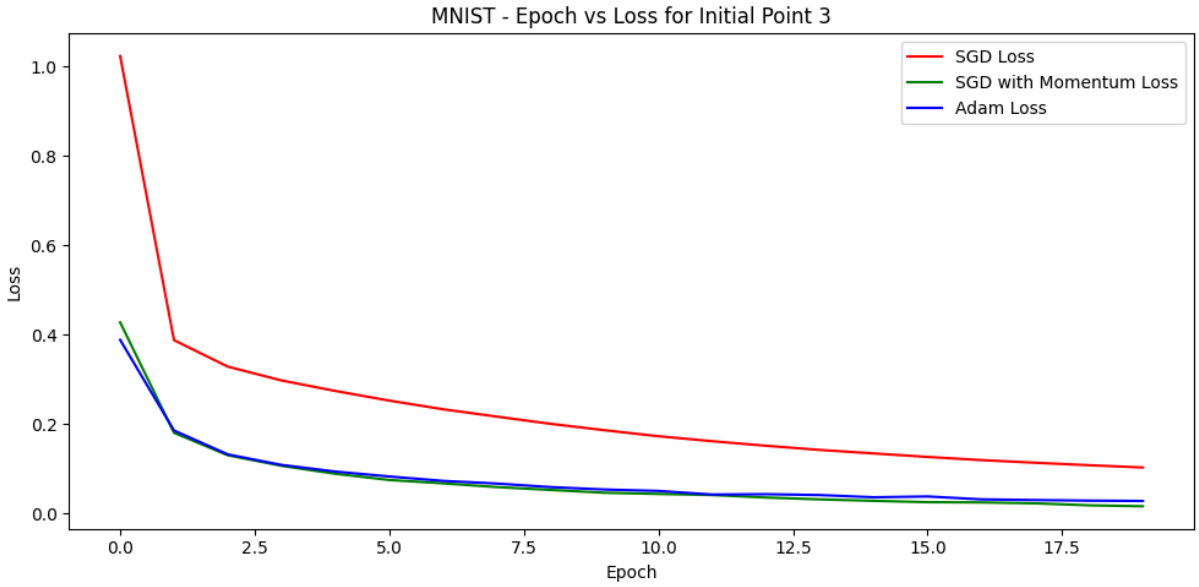
Yeşil kesikli çizgi, SGD with Momentum algoritmasını göstermektedir. Zamanla kayıp değeri düzenli bir şekilde azalmış ve yaklaşık 65 saniyede en düşük seviyeye ulaşmıştır. Momentum terimi sayesinde, SGD with Momentum algoritması zaman açısından daha verimli bir optimizasyon sunmuş ve daha kararlı bir şekilde kayıp değerini azaltmıştır.

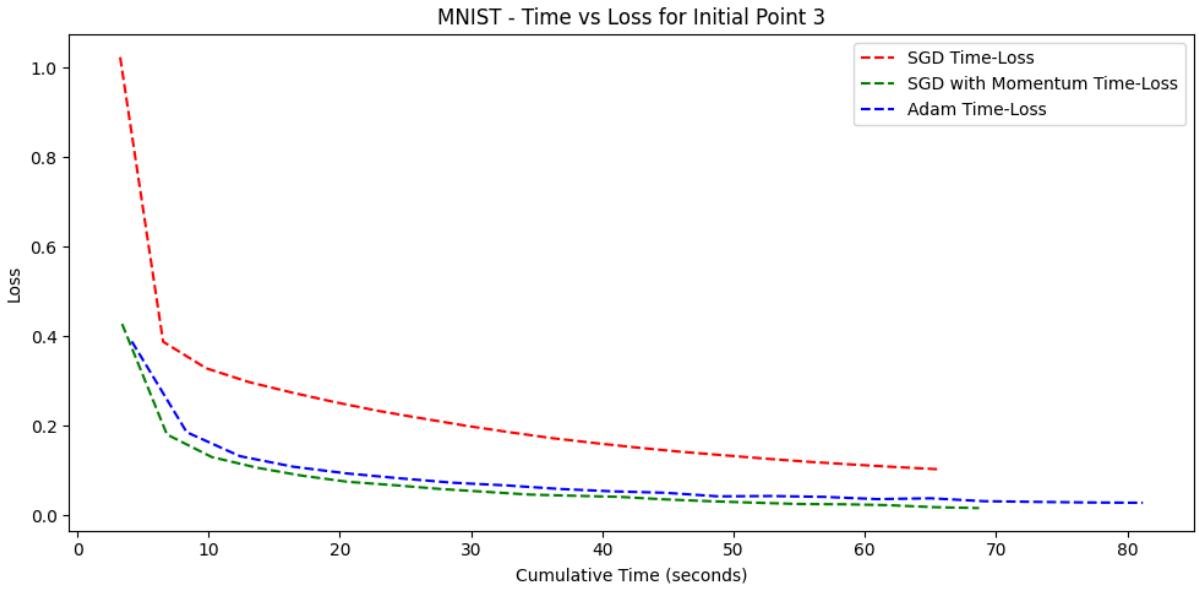
Mavi kesikli çizgi, Adam algoritmasını temsil etmektedir. Adam, özellikle ilk 10 saniye içinde kayıp değerini hızla düşürerek diğer algoritmalardan daha hızlı bir yakınsama göstermiştir. Zaman açısından en verimli algoritma olarak öne çıkmıştır. Toplam optimizasyon süresi boyunca en düşük kayıp değerine ulaşmıştır.

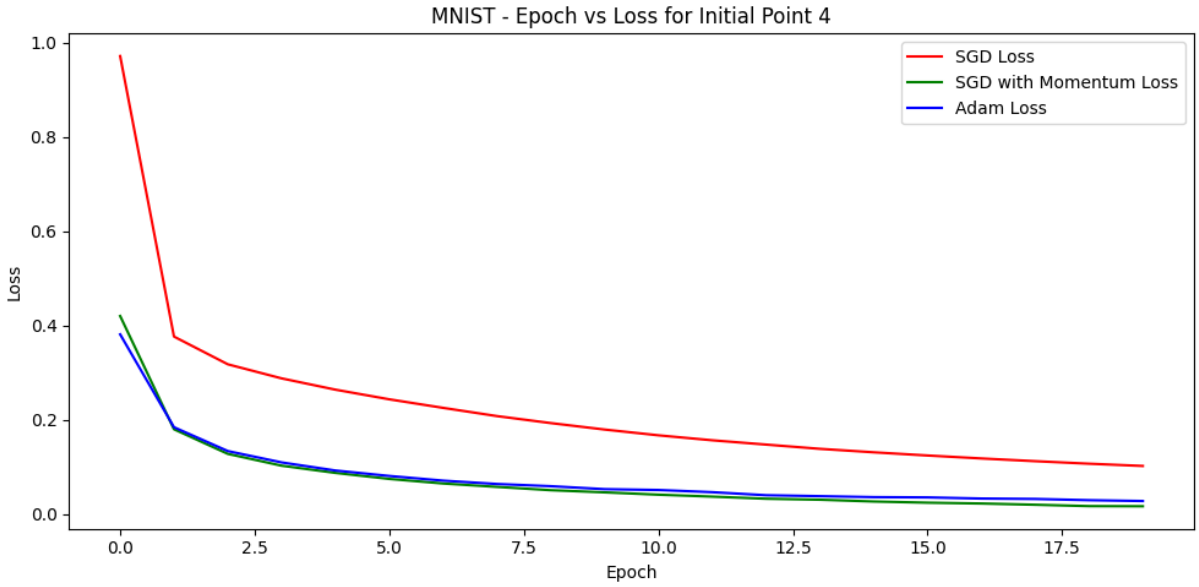
Diğer başlangıç noktalarına ait grafikler ise aşağıda sıralanmıştır.

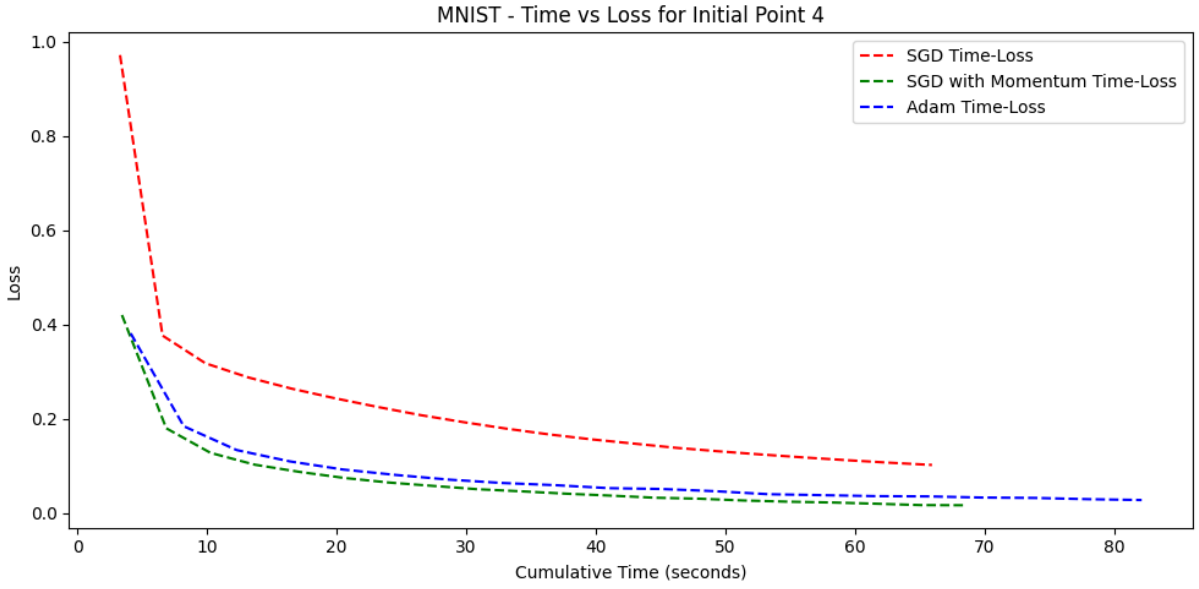


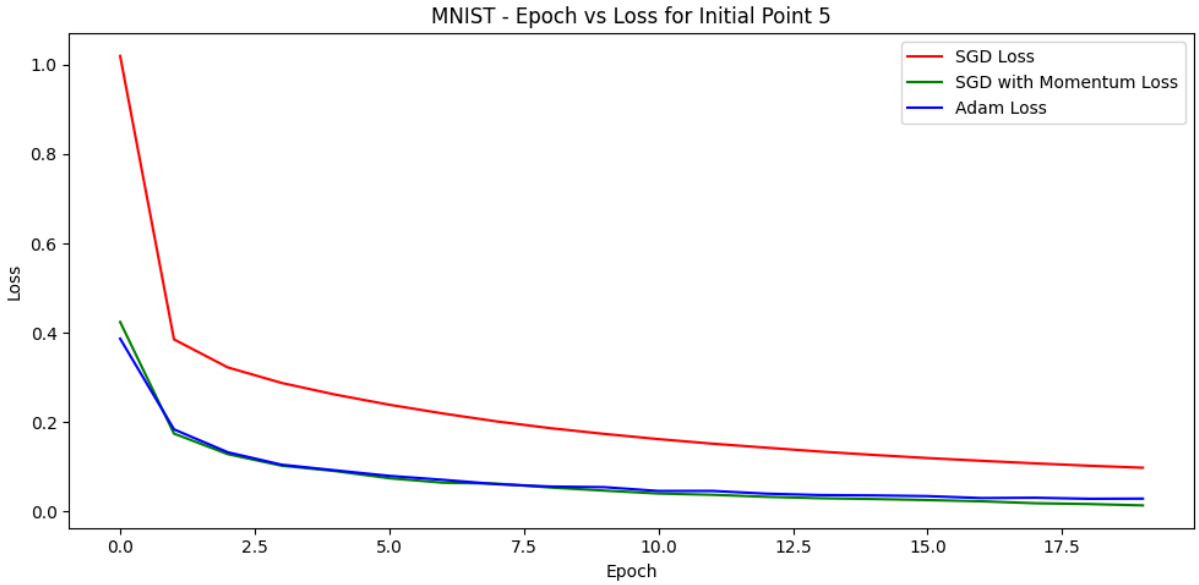


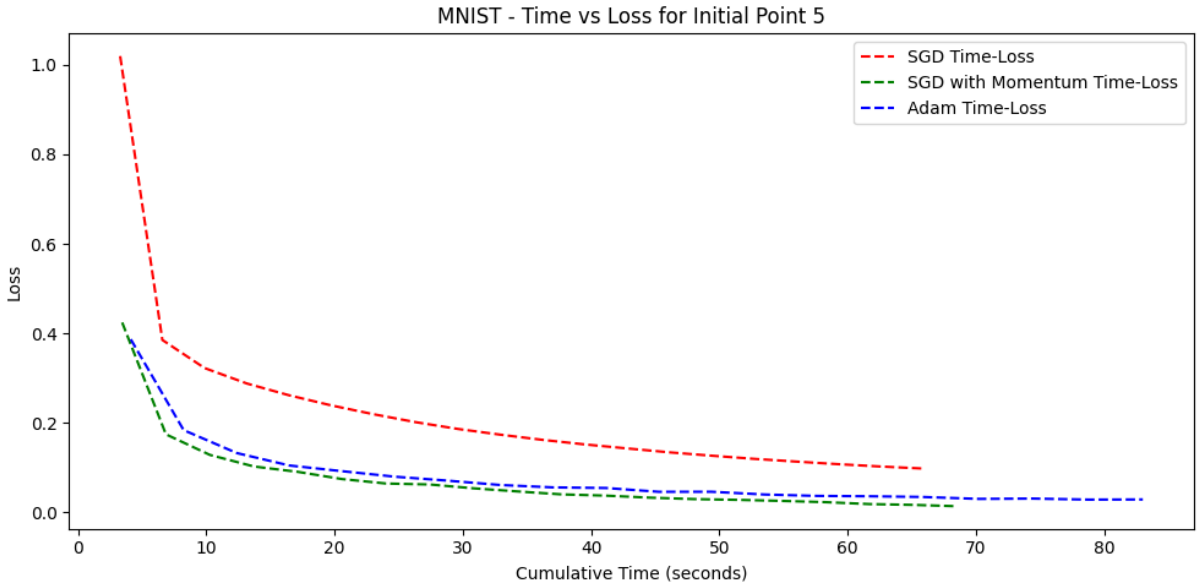

MNIST - Epoch vs Loss for Initial Point 3
SGD Loss
SGD with Momentum Loss
Adam Loss
Loss
Epoch
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


MNIST - Time vs Loss for Initial Point 3
SGD Time-Loss
SGD with Momentum Time-Loss
Adam Time-Loss
Loss
Cumulative Time (seconds)
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
60
70
80

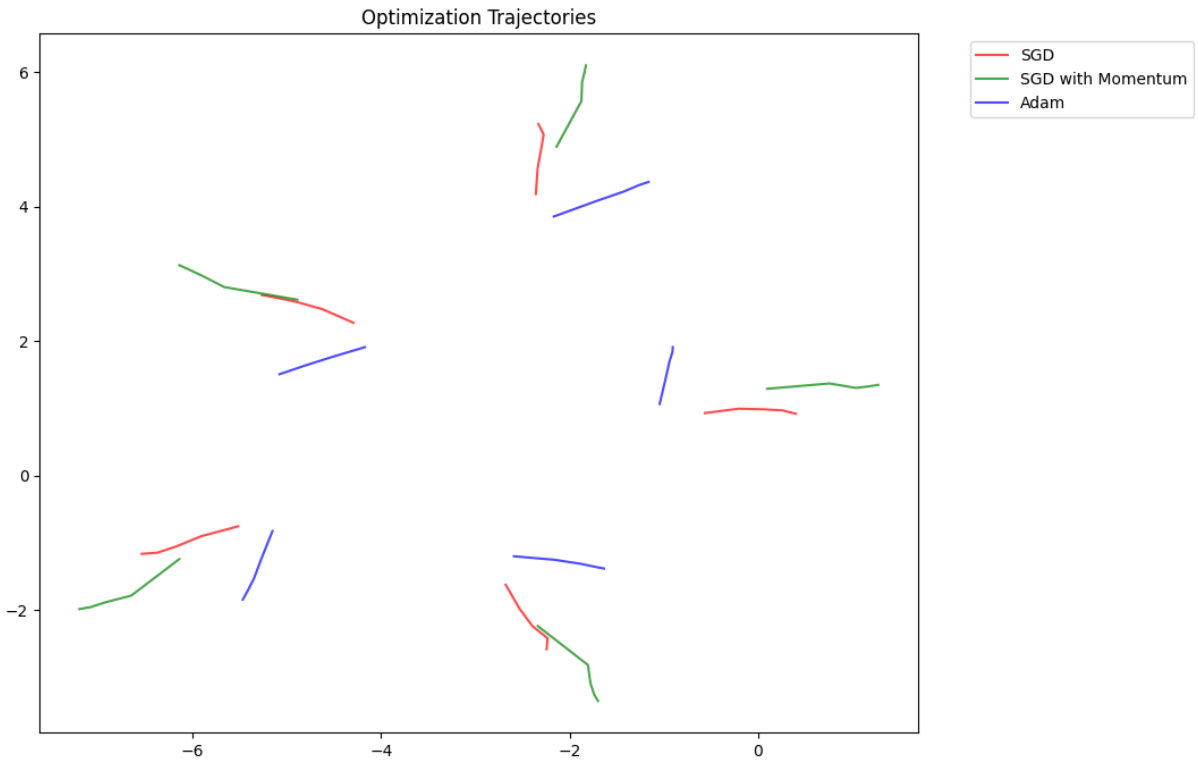

MNIST - Epoch vs Loss for Initial Point 4
SGD Loss
SGD with Momentum Loss
Adam Loss
Loss
Epoch
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


MNIST - Time vs Loss for Initial Point 4
SGD Time-Loss
SGD with Momentum Time-Loss
Adam Time-Loss
Loss
Cumulative Time (seconds)
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
60
70
80


MNIST - Epoch vs Loss for Initial Point 5
SGD Loss
SGD with Momentum Loss
Adam Loss
Loss
Epoch
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


MNIST - Time vs Loss for Initial Point 5
SGD Time-Loss
SGD with Momentum Time-Loss
Adam Time-Loss
Loss
Cumulative Time (seconds)
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
60
70
80

Aşağıdaki grafikte, t-SNE yöntemi kullanılarak farklı başlangıç noktalarından başlayan üç optimizasyon algoritmasının optimizasyon trajektoryaları görselleştirilmiştir.



Her algoritma, aynı başlangıç noktasından başlayarak parametre uzayındaki değişimlerini temsil eden yollar çizmektedir. Kırmızı çizgiler SGD algoritmasını, yeşil çizgiler SGD with Momentum algoritmasını ve mavi çizgiler ise Adam algoritmasını temsil etmektedir. Grafikte beş farklı başlangıç noktası için her algoritmanın trajektoryası gösterilmektedir.

Adam algoritması, adaptif öğrenme oranı sayesinde diğer algoritmalara kıyasla daha kısa ve stabil trajektoryalar çizmiştir. Bu durum, Adam algoritmasının daha az osilasyonla parametre uzayında hızlı bir şekilde yakınsama sağladığını ve kararlı hareket ettiğini göstermektedir. Başlangıç noktalarına duyarlılığı düşük olan Adam algoritması, farklı başlangıç noktalarında dahi benzer trajektoryalar sergileyerek verimli bir optimizasyon sağlamıştır.

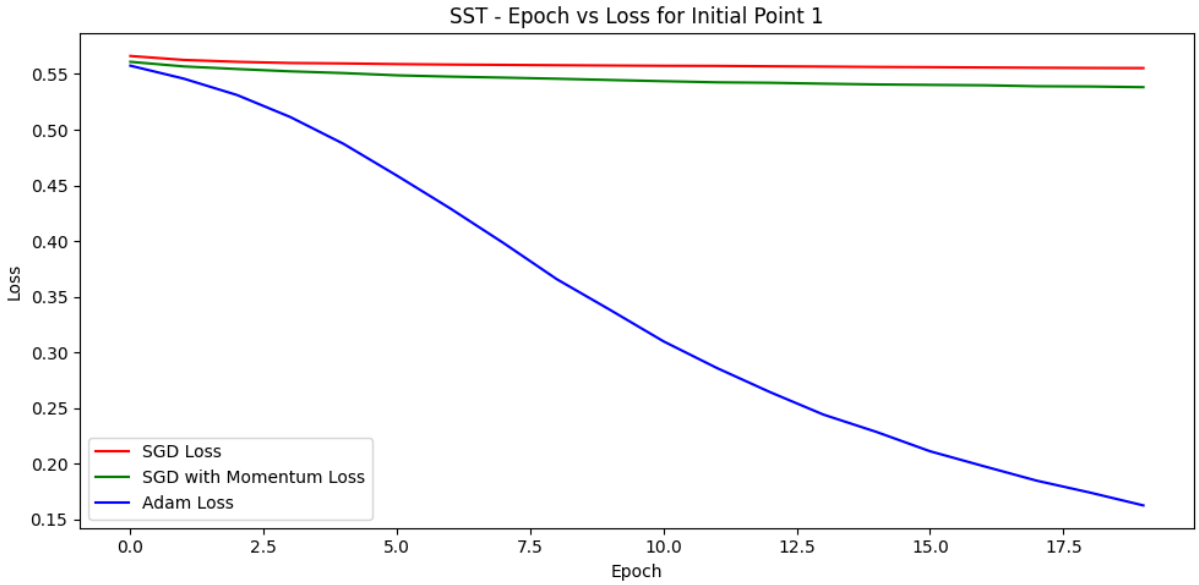
SGD algoritması, daha uzun ve düzensiz trajektoryalar çizmiş, başlangıç noktalarına olan duyarlılığı diğer algoritmalara göre daha belirgin olmuştur. Sabit öğrenme oranı kullanması nedeniyle, parametre uzayında daha fazla osilasyon yaşamış ve daha yavaş bir yakınsama süreci göstermiştir. Buna karşın, SGD with Momentum algoritması, momentum teriminin etkisiyle daha düzgün ve kararlı trajektoryalar çizmiştir. Momentum, osilasyonları azaltarak daha hızlı ve stabil bir yakınsama sağlamış, ancak Adam algoritması kadar hızlı bir şekilde en düşük kayıp değerine ulaşamamıştır.

Başlangıç noktalarının etkisi, algoritmalar arasında belirgin farklılıklar yaratmıştır. Adam algoritması başlangıç noktalarına daha az duyarlı iken, SGD ve SGD with Momentum algoritmalarında başlangıç noktalarının optimizasyon sürecini daha fazla etkilediği

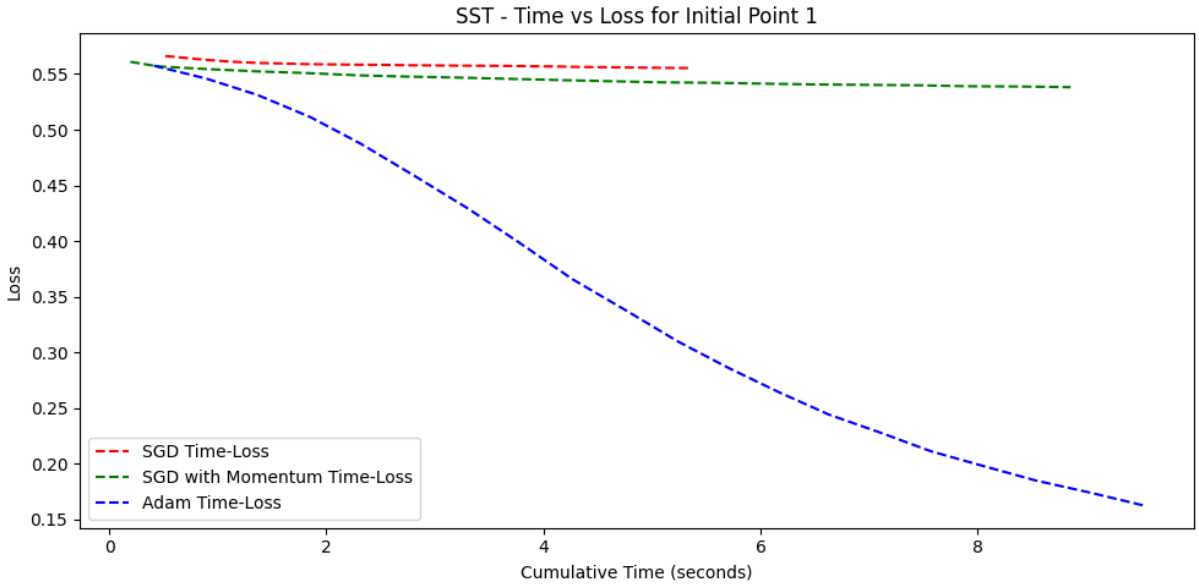
gözlemlenmiştir. SGD'nin daha uzun ve karmaşık trajektoryalar çizmesi, başlangıç noktasına olan duyarlılığını artırırken, SGD with Momentum bu etkiyi bir ölçüde dengelemiştir.

5.2 SST Veri Seti

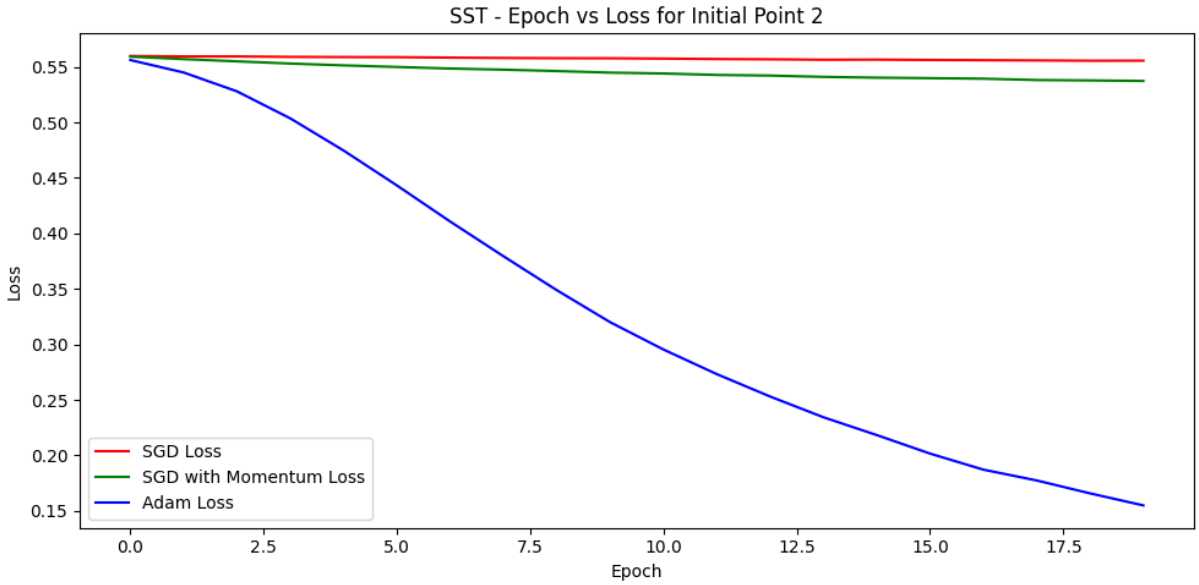
SST veri seti üzerinde yapılan deneyde, her bir noktadan başlayan algoritmaların epoch-loss ve time-loss grafikleri aşağıdaki gibi verilmiştir.

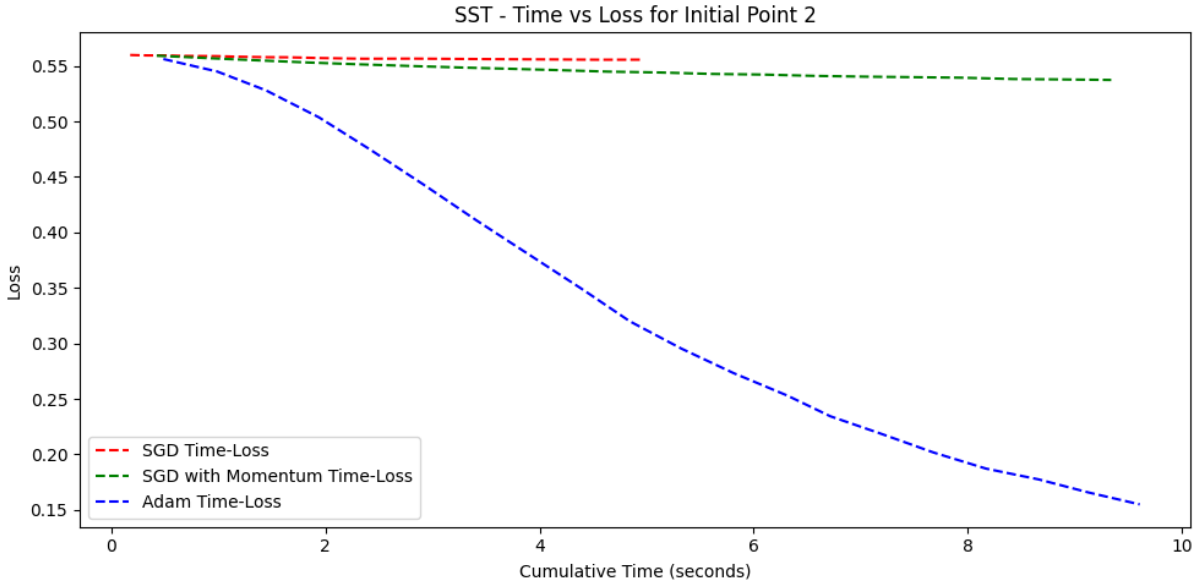


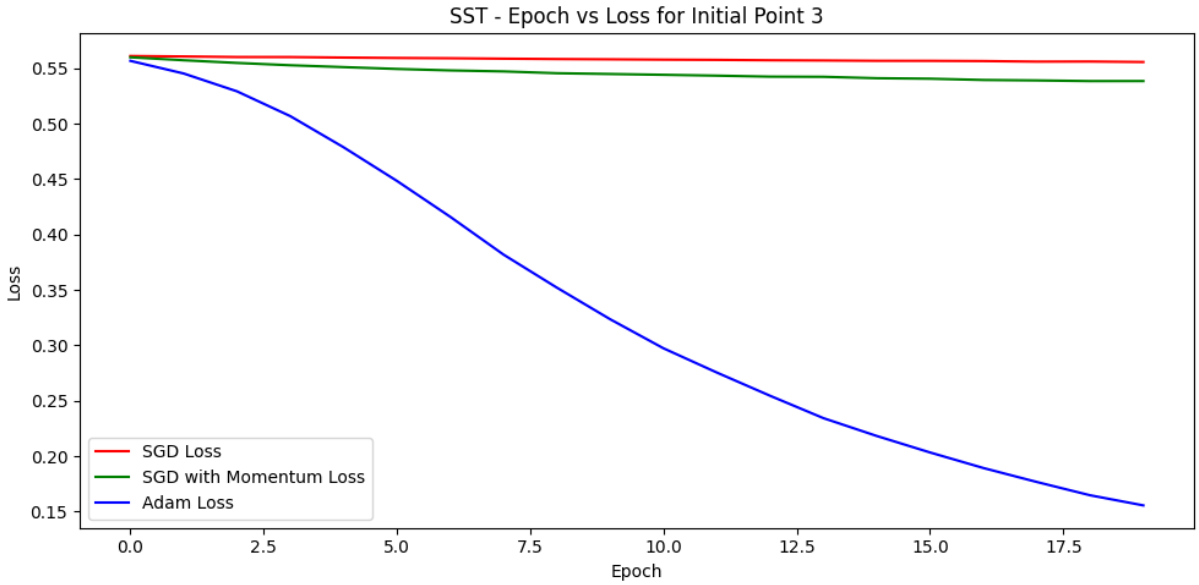
Bu grafik, SST veri seti üzerinde yapılan optimizasyon sürecinde epoch başına kayıp değerindeki değişimi göstermektedir. Burada üç optimizasyon algoritmasının performansı karşılaştırılmaktadır: SGD, SGD with Momentum ve Adam. Grafik, Adam algoritmasının açık bir şekilde daha iyi bir yakınsama sağladığını ortaya koymaktadır. Mavi çizgiyle temsil edilen Adam, her epoch'ta kayıp değerini düzenli bir şekilde azaltmış ve 20. epoch'a kadar en düşük kayıp değerine ulaşmıştır. SGD with Momentum (yeşil çizgi), kayıp değerini azaltmada SGD'ye (kırmızı çizgi) göre daha iyi performans göstermiştir. Ancak her iki algoritma da Adam'ın performansından uzak kalmıştır. SGD algoritması, sabit bir öğrenme oranı kullandığından dolayı kayıp değerindeki azalma oldukça sınırlı olmuş ve yakınsama sağlamakta başarısız kalmıştır. SGD with Momentum, momentum teriminin etkisiyle daha kararlı bir optimizasyon süreci sergilemiş ancak epoch sayısı arttıkça performansı stabilize olmuştur. Adam'ın adaptif öğrenme oranı, kayıp fonksiyonunda daha hızlı ve etkili bir şekilde azalma sağlamış, özellikle ilk birkaç epoch boyunca diğer algoritmalara kıyasla çok daha hızlı bir ilerleme göstermiştir.

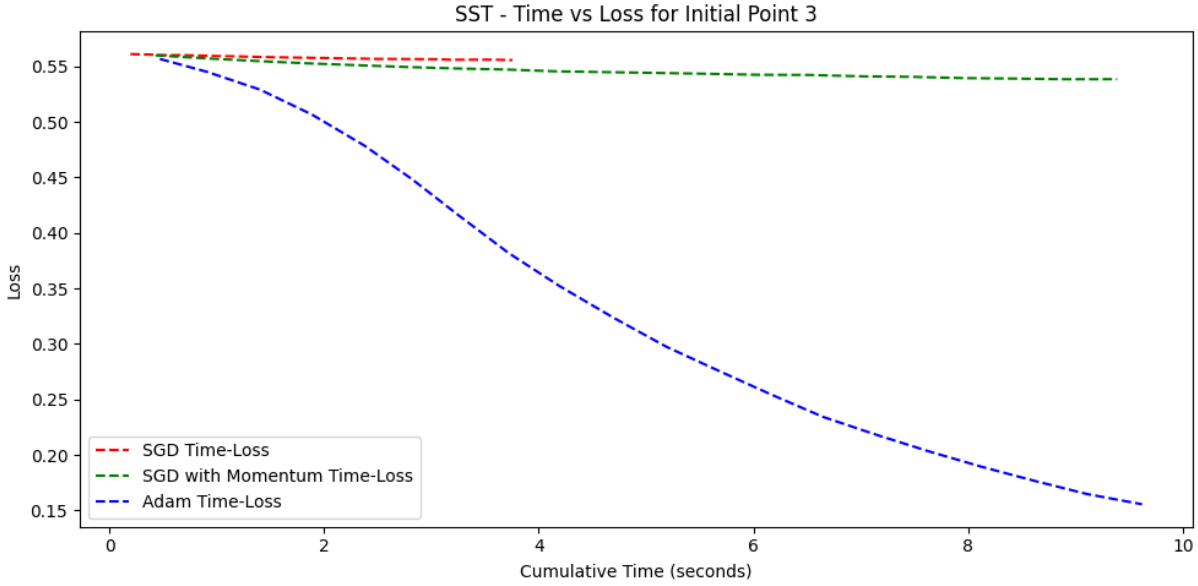


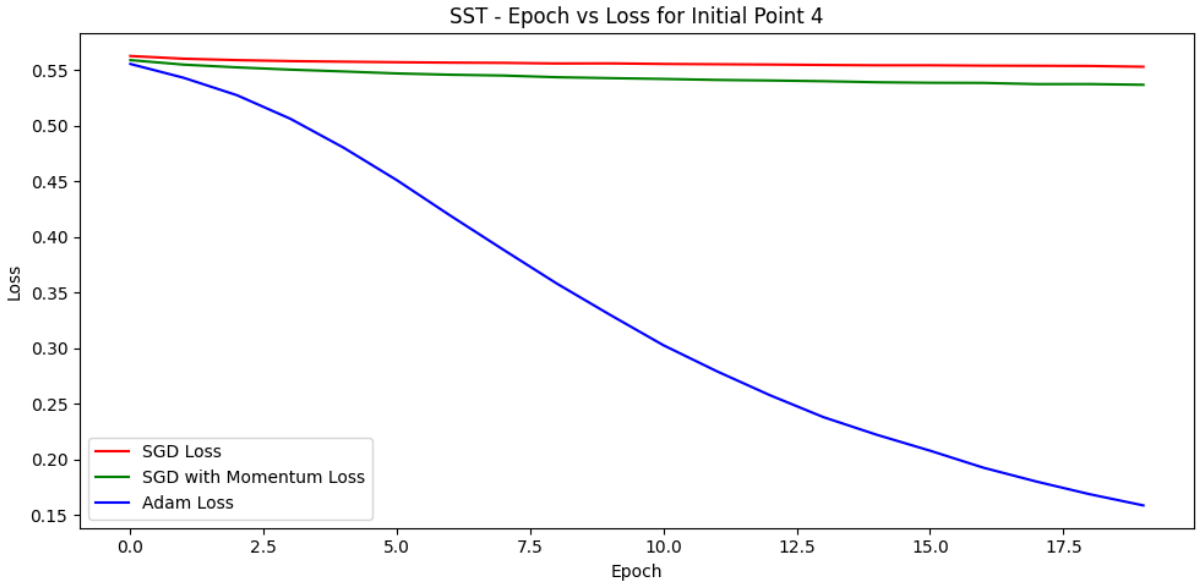
Bu grafik, zaman bazında kayıp değerindeki değişimi göstermektedir. Adam algoritması, hem hız hem de verimlilik açısından üstün performans sergilemiştir. Grafik, Adam algoritmasının çok kısa bir süre içinde kayıp değerini önemli ölçüde azalttığını ve en düşük kayıp değerine ulaştığını göstermektedir. SGD ve SGD with Momentum algoritmaları, zamanla kayıp değerlerini stabilize etmiş ancak Adam'a kıyasla daha yüksek kayıp seviyelerinde kalmıştır. SGD'nin kayıp değeri, optimizasyon süresi boyunca neredeyse sabit kalırken, SGD with Momentum bir miktar azalma sağlamış ancak yeterince etkili bir yakınsama gösterememiştir. Adam algoritmasının bu üstün performansı, adaptif öğrenme oranlarının, özellikle yüksek boyutlu ve karmaşık veri setlerinde daha verimli bir optimizasyon sağladığını göstermektedir. Ayrıca Adam, hem zaman hem de epoch bazında diğer algoritmalara kıyasla daha hızlı bir yakınsama sağlamış ve optimizasyon sürecinde en etkili algoritma olarak öne çıkmıştır. Bu sonuçlar, SST gibi metin tabanlı veri setlerinde Adam'ın diğer algoritmalara kıyasla daha verimli çalıştığını ve daha düşük kayıp değerlerine ulaştığını ortaya koymaktadır.

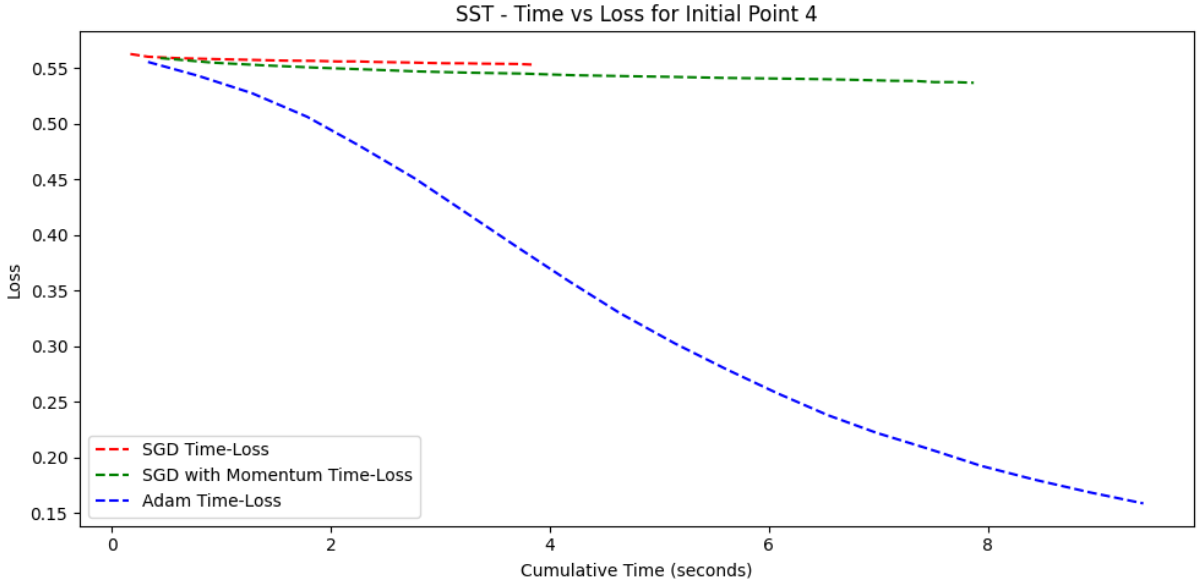

SST - Epoch vs Loss for Initial Point 2
Loss
Epoch
SGD Loss
SGD with Momentum Loss
Adam Loss
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

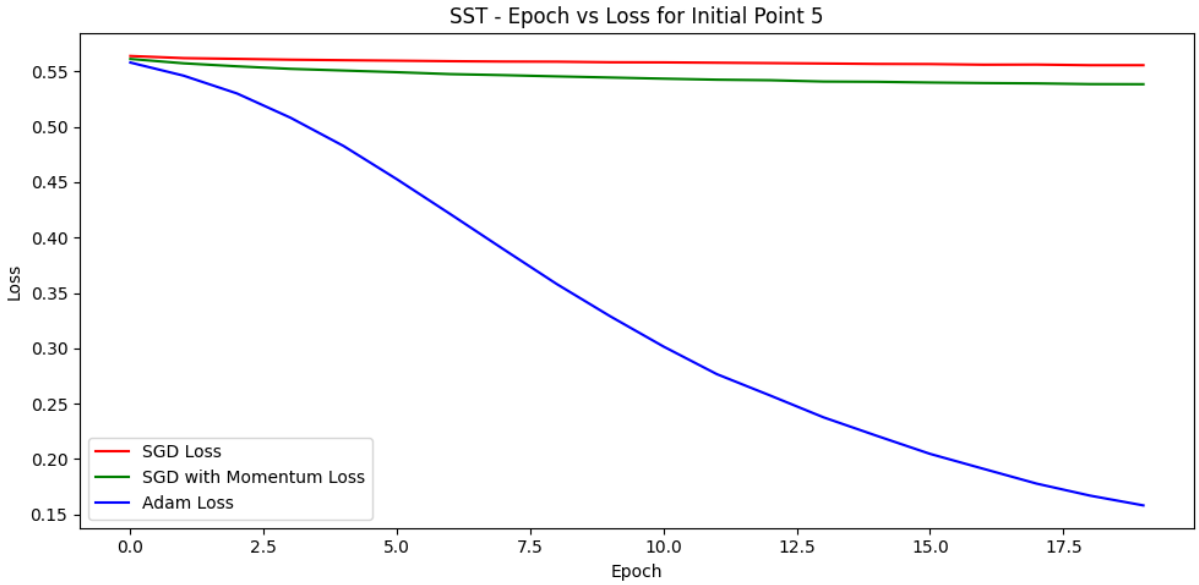

SST - Time vs Loss for Initial Point 2
Loss
Cumulative Time (seconds)
SGD Time-Loss
SGD with Momentum Time-Loss
Adam Time-Loss
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0
2
4
6
8
10

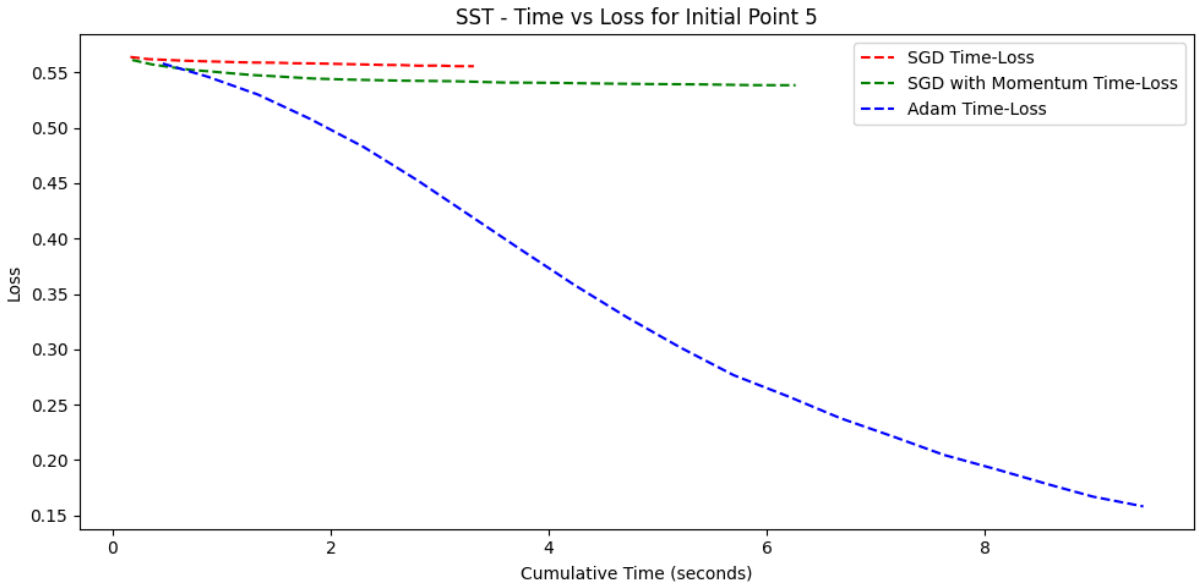

SST - Epoch vs Loss for Initial Point 3
Loss
Epoch
SGD Loss
SGD with Momentum Loss
Adam Loss
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

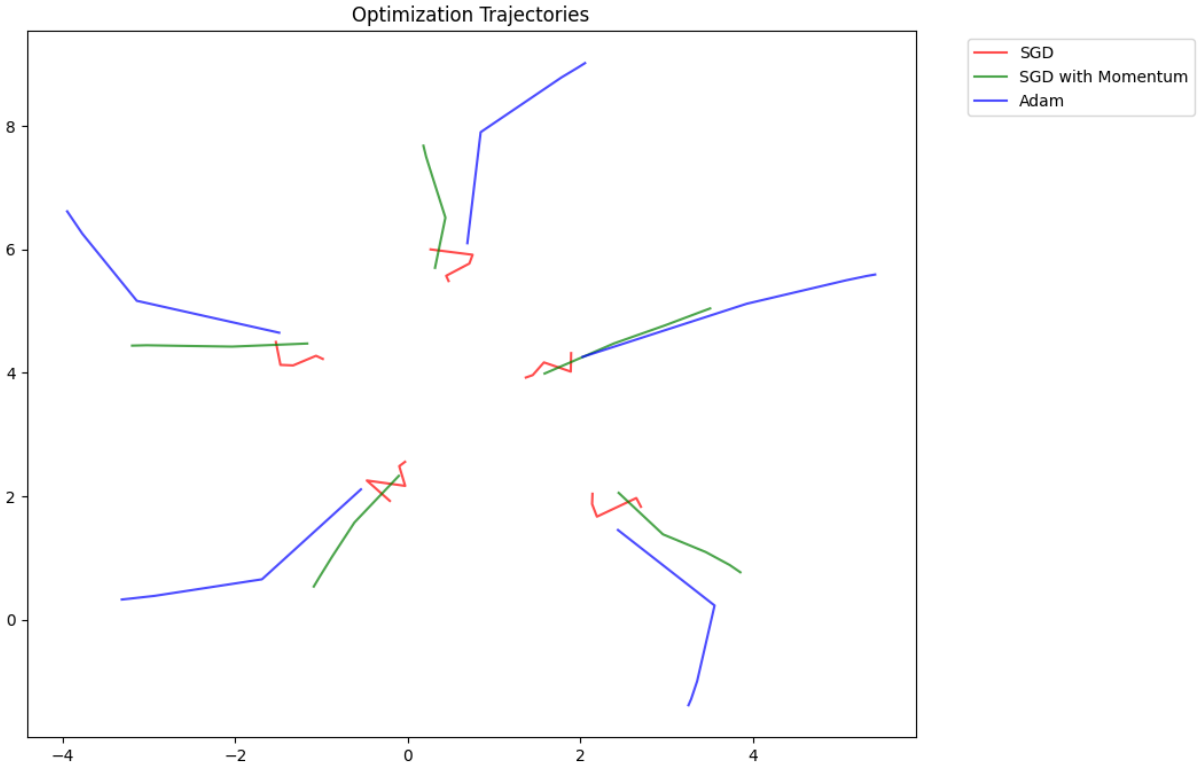

SST - Time vs Loss for Initial Point 3
Loss
Cumulative Time (seconds)
SGD Time-Loss
SGD with Momentum Time-Loss
Adam Time-Loss
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0
2
4
6
8
10


SST - Epoch vs Loss for Initial Point 4
Loss
Epoch
SGD Loss
SGD with Momentum Loss
Adam Loss
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


SST - Time vs Loss for Initial Point 4
Loss
Cumulative Time (seconds)
SGD Time-Loss
SGD with Momentum Time-Loss
Adam Time-Loss
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0
2
4
6
8


SST - Epoch vs Loss for Initial Point 5
Loss
Epoch
SGD Loss
SGD with Momentum Loss
Adam Loss
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


SST - Time vs Loss for Initial Point 5
Loss
Cumulative Time (seconds)
SGD Time-Loss
SGD with Momentum Time-Loss
Adam Time-Loss
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0
2
4
6
8



Bu grafik, SST veri seti için farklı başlangıç noktalarından başlayan üç optimizasyon algoritmasının parametre uzayındaki trajektoryalarını t-SNE kullanarak görselleştirmektedir. Her bir algoritma için çizilen yollar, optimizasyon sürecinde parametrelerin nasıl değiştiğini ve algoritmaların yakınsama hızlarını ortaya koymaktadır. Farklı başlangıç noktaları, optimizasyon sürecinde algoritmaların hareket ettiği uzayın çeşitli bölgelerini temsil etmektedir. Bu sayede algoritmaların başlangıç noktasına duyarlılığı ve parametre uzayındaki davranışları karşılaştırılabilir hale gelmiştir.

Adam algoritması, grafikte mavi çizgilerle temsil edilmektedir ve diğer algoritmalara kıyasla daha uzun ve genellikle kararlı bir yol izlemektedir. Bu durum, Adam'ın parametre uzayında daha büyük adımlarla hareket ettiğini ve başlangıç noktasına daha az duyarlı olduğunu göstermektedir. Ayrıca Adam'ın trajektoryalarının genellikle düz ve daha kararlı bir şekilde ilerlemesi, adaptif öğrenme oranlarının etkisiyle daha kararlı bir optimizasyon sağladığını ve osilasyonların minimize edildiğini göstermektedir. Adam algoritması, optimizasyon sürecinde genellikle daha uzun bir yol izlese de bu yolun stabil olması, algoritmanın etkili bir şekilde parametre uzayında gezinmesini sağlamaktadır.

SGD algoritması ise kırmızı çizgilerle temsil edilmiştir. SGD'nin trajektoryaları genellikle kısa ve düzensizdir, bu da algoritmanın yakınsamada zorluk çektiğini ve başlangıç noktasına oldukça duyarlı olduğunu göstermektedir. Parametre uzayında osilasyonlara daha açık olan SGD, optimizasyon sürecinde daha rastgele bir yol izlemektedir. Bu durum, sabit öğrenme oranı kullanımı ve momentum teriminin eksikliğinden kaynaklanmaktadır. SGD'nin başlangıç noktası seçimine bağlı olarak farklı trajektoryalar çizmesi, algoritmanın istikrarsız bir yapıya sahip olduğunu göstermektedir.

SGD with Momentum algoritması yeşil çizgilerle temsil edilmiştir ve SGD'ye kıyasla daha uzun ve düzenli trajektoryalar çizmektedir. Momentum terimi, osilasyonları azaltarak algoritmanın daha kararlı bir şekilde ilerlemesini sağlamıştır. Ancak SGD with Momentum'un trajektoryaları, Adam kadar düz ve istikrarlı değildir. Bunun nedeni, momentumun optimizasyon sürecinde yalnızca geçmiş gradyanlara bağlı bir düzeltme sağlaması, ancak adaptif bir öğrenme oranı mekanizmasının olmamasıdır. Bu nedenle SGD with Momentum, Adam'a kıyasla daha az kararlı bir yol izlemiş ve başlangıç noktasına duyarlılığı hala belirgin bir şekilde devam etmiştir.

6. SONUÇ

Bu çalışmada, Stokastik Gradient Descent (SGD), SGD with Momentum ve Adam olmak üzere üç farklı optimizasyon algoritmasının MNIST ve SST veri setleri üzerindeki performansları karşılaştırılmıştır. Görüntü ve metin tabanlı veri setleri kullanılarak, algoritmaların parametre uzayındaki trajektoryaları t-SNE ile görselleştirilmiş, epoch ve zaman bazlı analizlerle performansları değerlendirilmiştir. Elde edilen sonuçlar, algoritmaların her biri için belirgin farklılıkları ve güçlü yönleri ortaya koymaktadır.

Adam algoritması, adaptif öğrenme oranı mekanizması sayesinde hem epoch hem de zaman bazlı analizlerde en hızlı ve kararlı yakınsamayı sağlamıştır. İlk birkaç iterasyonda kayıp değerlerini hızla azaltarak diğer algoritmalardan belirgin şekilde ayrılmış ve her iki veri setinde de en düşük kayıp değerine ulaşmıştır. Bu durum, Adam algoritmasının özellikle karmaşık ve yüksek boyutlu veri setlerinde etkili bir çözüm sunduğunu göstermektedir.

SGD algoritması, sabit öğrenme oranı kullanımı nedeniyle daha yavaş bir yakınsama göstermiş ve osilasyonlara daha açık olmuştur. Momentum teriminin eklendiği SGD with Momentum algoritması, SGD'ye kıyasla daha kararlı bir performans sergilemiş ve kayıp değerlerini daha hızlı bir şekilde azaltmıştır. Ancak, SGD with Momentum'un performansı, Adam algoritmasının adaptif yaklaşımıyla kıyaslandığında sınırlı kalmıştır.

Trajektorya görselleştirmeleri, farklı başlangıç noktalarının algoritmaların yakınsama süreçlerini nasıl etkilediğini açıkça ortaya koymuştur. Adam algoritmasının başlangıç noktalarına daha az duyarlı olduğu, SGD'nin ise başlangıç noktasına oldukça duyarlı bir şekilde davranarak daha düzensiz ve yavaş bir optimizasyon süreci izlediği gözlemlenmiştir. SGD with Momentum ise bu iki algoritma arasında bir denge sağlamış ve daha düzenli bir optimizasyon süreci sunmuştur.

7. KAYNAKLAR

1. Kingma, D. P., & Ba, J. (2014). "Adam: A Method for Stochastic Optimization." *arXiv preprint arXiv:1412.6980*

2. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11), 2278–2324.

3. Sachinsoni. (n.d.). *Mastering t-SNE(t-distributed stochastic neighbor embedding)*. Medium. https://medium.com/@sachinsoni600517/mastering-t-sne-t-distributed-stochastic-neighbor-embedding-0e365ee898ea.

4. Van der Maaten, L., & Hinton, G. (2008). "Visualizing data using t-SNE." *Journal of Machine Learning Research*, 9(Nov), 2579-2605.