# TENSORFLOW FREYMVORKI:O‘RNATISH VA ASOSIY FUNKSIYALAR

**Tojimamatov Israil Nurmamatovich**
*Farg‘ona davlat universiteti Amaliy matematika va informatika kafedrasi katta o’qituvchisi*
*E-mail: israiltojimamatov@gmail.com*
**Mamadaliyeva Erkinoy Lutfullo qizi**
*Farg‘ona davlat universiteti Amaliy matematika yo‘nalishi 3-bosqich 25.09-guruh talabasi*
*E-mail: mamadaliyevaerkinoy@gmail.com*

**Annotatsiya**: *Mazkur ilmiy ish zamonaviy chuqur o‘rganish (Deep Learning) sohasidagi eng yetakchi dasturiy ta’minot vositalaridan biri bo‘lgan TensorFlow freymvorkining kompleks tahliliga bag‘ishlangan. Maqolada freymvorkni turli operatsion tizimlarda muvaffaqiyatli o‘rnatish jarayonlari, jumladan, virtual muhitlarni sozlashning nozik jihatlari batafsil ko‘rib chiqiladi. Shuningdek, TensorFlowning ichki me’morchiligi, xususan, statik hisoblash graflaridan tortib to Eager Execution (tezkor ijro) modeligacha bo‘lgan evolyutsiyasi chuqur o‘rganiladi. Asosiy funksionallik qismi esa neyron tarmoqlarning turli turlarini yaratish, ma’lumotlar bilan ishlash uchun tf.data API imkoniyatlari hamda o‘qitilgan modellarni saqlash va joylashtirish mexanizmlarini o‘z ichiga oladi. Ushbu tadqiqot o‘qituvchilar, magistrantlar va soha mutaxassislari uchun chuqur o‘rganish loyihalarini professional darajada amalga oshirishda fundamental bilim manbai bo‘lib xizmat qiladi.*

**Kalit So‘zlar***: TensorFlow, Chuqur O‘rganish, Mashinani O‘rganish, Neyron Tarmoqlar, Dasturiy Me’morchilik, Freymvork O‘rnatish.*

**Аннотация***: Настоящая научная статья посвящена всестороннему анализу фреймворка TensorFlow, одного из ключевых инструментов в области современного глубокого обучения (Deep Learning). В работе подробно рассматриваются процессы успешной установки фреймворка на различных операционных системах, включая тонкости настройки виртуальных сред. Особое внимание уделяется глубокому изучению внутренней архитектуры TensorFlow, начиная со статических графов вычислений и заканчивая эволюцией к модели Eager Execution (немедленное выполнение). Раздел, посвященный основной функциональности, охватывает создание различных типов нейронных сетей, возможности API tf.data для эффективной работы с большими объемами данных, а также механизмы сохранения и развертывания обученных моделей. Данное исследование призвано стать фундаментальным источником знаний для преподавателей, магистрантов и специалистов в данной области, стремящихся к профессиональной реализации проектов по глубокому обучению.*

**Ключевые Слова***: TensorFlow, Глубокое Обучение, Машинное Обучение, Нейронные Сети, Архитектура Программного Обеспечения, Установка Фреймворка.*

**Abstract***: This scientific paper provides a comprehensive analysis of the TensorFlow framework, one of the leading software tools in the field of modern Deep Learning. The study details the processes for successful installation of the framework across various operating systems, including the nuances of configuring virtual environments. Significant attention is paid to a deep exploration of TensorFlow's internal architecture, tracing its evolution from static computation graphs to the dynamic Eager Execution model. The core functionality section covers the creation of different types of neural networks, the capabilities of the tf.data API for efficient data handling, and the mechanisms for saving and deploying trained models. This research is intended to serve as a fundamental resource for educators, graduate students, and industry professionals seeking to implement Deep Learning projects at a high professional level.*

**Key Words***: TensorFlow, Deep Learning, Machine Learning, Neural Networks, Software Architecture, Framework Installation.*

**KIRISH**

Zamonaviy kompyuter fanlari va sun'iy intellekt (AI) texnologiyalari sohasidagi so'nggi yutuqlar dunyo bo'ylab ilmiy va sanoat paradigmalarni tubdan o'zgartirib yubordi. Ushbu inqilobning markazida chuqur o'rganish (Deep Learning) deb ataluvchi mashinani o'rganishning bir tarmog'i turadi. Chuqur o'rganish modellari ma'lumotlarning murakkab ierarxik xususiyatlarini avtomatik ravishda o'rganish qobiliyatiga ega bo'lganligi sababli, ular tasvirni aniqlash, tabiiy tilni qayta ishlash, nutqni tanish va hattoki avtonom boshqaruv tizimlari kabi sohalarda insoniyat imkoniyatlariga yaqin natijalarni namoyish etmoqdalar.

Biroq, bunday ulkan hisoblash quvvatini va murakkab me'moriy tuzilmalarni amaliyotga tatbiq etish uchun samarali, moslashuvchan va kuchli dasturiy ta'minot vositalariga ehtiyoj paydo bo'ldi. Aynan shu nuqtada mashinani o'rganish freymvorklari, xususan TensorFlow, muhim ahamiyat kasb etadi. Bu freymvorklar tadqiqotchilar va muhandislarga ulkan ma'lumotlar to'plamlari ustida ishlash imkonini beruvchi hisoblash graflarini samarali qurish, optimallashtirish va tezlashtirilgan qurilmalar (masalan, grafik protsessorlar yoki maxsus TPU'lar) ustida ijro etish uchun zarur bo'lgan barcha qurollarni taqdim etadi.

TensorFlow, Google Brain jamoasi tomonidan ishlab chiqilgan va ochiq manbali platforma sifatida jamoatchilikka taqdim etilgan freymvorkdir. Uning asosiy maqsadi – neyron tarmoqlarning murakkab matematik operatsiyalarini o'z ichiga olgan hisoblash jarayonlarini tensorlar (ko'p o'lchovli ma'lumotlar massivlari) oqimi sifatida ifodalash. Ushbu freymvorkning dastlabki yaratilishi katta hajmdagi ma'lumotlarni qayta ishlash va global miqyosdagi o'rganish modellarini yaratishga qaratilgan edi. Uning moslashuvchan me'morchiligi tadqiqot muhitidan tortib, mobil qurilmalar va brauzerlar kabi cheklangan

resursli platformalarda modellarni joylashtirishgacha bo'lgan keng doiradagi amaliyotlarni qo'llab-quvvatlaydi.

Ushbu ilmiy tadqiqotning asosiy maqsadi chuqur o'rganish sohasida faoliyat yuritayotgan o'qituvchilar, magistrantlar va amaliyotchi mutaxassislar uchun TensorFlow freymvorki haqida chuqur, tizimli va professional tahlilni taqdim etishdan iborat. Biz freymvorkning shunchaki yuza o'rnatish jarayonlariga to'xtalmasdan, uning ichki me'moriy evovolutsiyasini tahlil qilamiz, asosiy funksionallikni Keras API yordamida chuqur ko'rib chiqamiz, shuningdek, ma'lumotlarni samarador boshqarish uchun tf.data kabi muhim yordamchi vositalarning amaliy ahamiyatini ochib beramiz.

Maqolaning tuzilishi o'quvchiga mavzuni silsilali ravishda tushunish imkonini beradi. Dastlab, biz TensorFlowning me'moriy asoslarini va uning tarixiy o'zgarishlarini yoritib beramiz. So'ngra, freymvorkni turli operatsion tizimlarda muvaffaqiyatli o'rnatishning barcha nozik jihatlari, xususan, grafik protsessorlar bilan integratsiya masalalarini atroflicha ko'rib chiqamiz. Maqolaning keyingi bo'limlari neyron tarmoqlarni yaratishning yuqori darajadagi usullari, ma'lumotlarni o'qish quvurini qurish va nihoyat, o'qitilgan modellarni baholash va amaliyotga joylashtirish mexanizmlariga bag'ishlanadi. Ushbu yondashuv tadqiqot materialining nafaqat nazariy asoslanganligini, balki chuqur o'rganish loyihalarini to'liq tsiklda amalga oshirish uchun zarur bo'lgan fundamental bilimlarni ham ta'minlashni kafolatlaydi.

TensorFlow freymvorkining amaliy qo'llanilishi uning operatsion tizimga muvaffaqiyatli va to'g'ri o'rnatilishidan boshlanadi. Bu bosqich nafaqat dasturiy ta'minotni kompyuterga yuklashni, balki loyihaning barqarorligini va ishlash samaradorligini ta'minlaydigan optimal dasturiy muhitni sozlashni ham o'z ichiga oladi.

TensorFlowni o'rnatishdan oldin, tizim ma'lum asosiy dasturiy va apparat talablariga javob berishi lozim. Dasturiy nuqtai nazardan, Python tilining mos versiyasi zarur hisoblanadi, chunki TensorFlow asosan Python ekotizimida faoliyat yuritadi. Muhim qism apparat talablariga tegishli bo'lib, ayniqsa grafik protsessorlar (GPY) yordamida tezlashtirilgan hisoblashlarni amalga oshirish ko'zda tutilganida bu talablar yanada qat'iylashadi. GPY bilan ishlash uchun Nvidia tomonidan taqdim etilgan maxsus dasturiy vositalar, xususan CUDA (Compute Unified Device Architecture) platformasi va cuDNN (CUDA Deep Neural Network) kutubxonalari talab etiladi. Ushbu komponentlarning bir-biriga mos keladigan versiyalari to'plamini sozlash yuqori samarali chuqur o'rganish loyihalari uchun fundamental ahamiyatga ega.

Professional va tizimli dasturiy ta'minotni ishlab chiqishda virtual muhitlardan foydalanish majburiy hisoblanadi. Virtual muhitlar (masalan, Conda yoki Pythonning o'zining venv moduli) har bir loyiha uchun alohida, izolyatsiya qilingan Python muhitini yaratish imkonini beradi. Bu izolyatsiya turli loyihalar orasidagi bog'liqliklar ziddiyatini oldini oladi, chunki har bir loyiha o'zining TensorFlow, NumPy yoki boshqa kutubxonalarning aniq versiyalarini talab qilishi mumkin. Loyihaning takrorlanuvchanligi (reproducibility) ham

virtual muhit yordamida ta'minlanadi, bu esa boshqa tadqiqotchilar uchun maqolada keltirilgan natijalarni qayta yaratish imkonini beradi.

TensorFlowni o'rnatishning eng keng tarqalgan usuli Pip paket boshqaruvchisi orqali amalga oshiriladi. Odatda, foydalanuvchilar ikki asosiy versiyadan birini tanlashi kerak bo'ladi: faqat markaziy protsessor (MPY) yordamida ishlaydigan standart versiya yoki GPY tomonidan tezlashtirilgan hisoblashlarni qo'llab-quvvatlaydigan versiya. GPY versiyasini o'rnatishda, yuqorida aytib o'tilgan CUDA va cuDNN drayverlari tizimda to'g'ri sozlangan bo'lishi shart. Versiyalarni boshqarishda, ayniqsa chuqur o'rganish sohasida, eng so'nggi barqaror versiyani ishlatish tavsiya etiladi, chunki freymvork doimiy ravishda optimallashtiriladi va yangilanadi.

TensorFlow yordamida murakkab neyron tarmoqlarni o'qitish grafik protsessorlarsiz amalda samarasizdir. GPY bilan integratsiya jarayoni faqat dasturiy ta'minotni o'rnatish bilan cheklanmaydi; u operatsion tizim darajasida apparat va drayverlarni to'g'ri sozlashni talab qiladi. Dastlab, GPY drayverlari yangilanishi, so'ngra TensorFlow kutubxonasi talab qiladigan CUDA Toolkit va cuDNN ning aniq versiyalari o'rnatilishi lozim. Ushbu kutubxonalar chuqur o'rganish uchun zarur bo'lgan matris operatsiyalarini va konvolyutsiya kabi fundamental jarayonlarni GPY ning parallel arxitekturasida juda yuqori tezlikda bajarish imkoniyatini beradi. Integratsiya muvaffaqiyatli bo'lsa, TensorFlow avtomatik ravishda mavjud GPY qurilmalarini aniqlaydi va hisoblashlarni optimallashtirish uchun ulardan foydalanadi, bu esa o'qitish vaqtini sezilarli darajada qisqartiradi.

Shunday qilib, TensorFlowni professional darajada o'rnatish, shunchaki buyruq qatorini ishga tushirish emas, balki chuqur muhit boshqaruvi, apparat va dasturiy ta'minot talablarini sinxronlashtirishni talab qiluvchi tizimli jarayondir. Bu barcha elementlar freymvorkning to'liq salohiyatini ochish uchun muhim hisoblanadi.

TensorFlow freymvorkining fundamental imkoniyatlarini chuqur tushunish uchun uning me'moriy asoslari va rivojlanishidagi asosiy bosqichlarni tahlil qilish zarur. Freymvorkning samaradorligi aynan uning ichki tuzilishiga va hisoblashlarni tashkil etish usuliga bog'liqdir.

Keras API'si barcha ushbu komponentlarni (qatlamlar, optimallashtiruvchilar va yo'qotish funksiyalari) bir joyga jamlab, tadqiqotchilarga chuqur o'rganish modellarini tezkor va tizimli ravishda qurish va tajriba o'tkazish uchun yagona va samarali interfeysni taqdim etadi.

TensorFlow freymvorkining amaliy qo'llanilishida uning asosiy funksionalligi, ayniqsa yuqori darajadagi abstraktsiyani ta'minlovchi Keras API orqali namoyon bo'ladi. Keras chuqur o'rganish modellarini yaratish, kompilyatsiya qilish va o'qitish jarayonlarini sezilarli darajada soddalashtirib, freymvorkning foydalanuvchilar orasida keng tarqalishiga xizmat qildi.

Keras dastlab alohida loyiha sifatida boshlangan bo'lsa-da, vaqt o'tishi bilan u TensorFlowning rasmiy va tavsiya etilgan yuqori darajadagi interfeysiga aylandi. Kerasning asosiy vazifasi foydalanuvchini past darajadagi hisoblash operatsiyalari va tensorlarni boshqarishning murakkabliklaridan xalos etib, e'tiborni modelning me'moriy dizayniga va tadqiqot muammosining yechimiga qaratishdir. Bu abstraktsiya yordamida tadqiqotchilar

neyron tarmoqlarning qatlamlarini (layers) oddiy, intuitiv kod yordamida bir-biriga ulash orqali murakkab modellarni tezda qurish imkoniyatiga ega bo'ladilar.

Model Turlari: Sequential va Functional API

Neyron tarmoq me'moriy tuzilmalarini yaratish uchun Keras ikki asosiy yondashuvni taqdim etadi, ular turli darajadagi moslashuvchanlikni taklif qiladi:

1. Sequential (Ketma-ket) Modeli: Bu yondashuv eng sodda bo'lib, bunda ma'lumotlar oqimi bir qatlamdan keyin ikkinchisiga to'g'ridan-to'g'ri o'tadigan oddiy, chiziqli stack (ustun) ko'rinishidagi tarmoqlarni yaratish uchun mo'ljallangan. U kirish qatlamidan chiqish qatlamigacha bir yo'nalishda harakatlanadigan klassik to'liq bog'langan tarmoqlar yoki oddiy konvolyutsion tarmoqlarni qurishda juda qulaydir.

2. Functional (Funksional) API: Bu yondashuv ancha moslashuvchan bo'lib, bitta kirish qatlamidan bir nechta qatlamga ma'lumot uzatilishi mumkin bo'lgan murakkab, chiziqli bo'lmagan me'moriy tuzilmalarni yaratish imkonini beradi. U turli manbalardan kelgan ma'lumotlarni birlashtiruvchi tarmoqlar, bir nechta chiqishga ega bo'lgan modellar yoki rezidual bog'lanishlar (residual connections) kabi ilg'or me'moriy elementlarni o'z ichiga olgan tarmoqlarni, masalan, ResNet yoki Inception modellarini yaratish uchun zarurdir. Funksional API neyron tarmoqning har bir qatlamini ma'lumotlarni qayta ishlaydigan funktsiya sifatida ko'rishga imkon beradi.

Qatlamlar neyron tarmoqning asosiy qurilish bloklari hisoblanadi. Har bir qatlam o'ziga kelgan tensor ma'lumotlari ustida aniq belgilangan matematik operatsiyalarni bajaradi va o'zgartirilgan tensorlarni keyingi qatlamga uzatadi.

Dense (To'liq Bog'langan) Qatlamlar: Bu qatlamdagi har bir neyron avvalgi qatlamdagi har bir neyron bilan bog'langan bo'ladi. Ular ma'lumotlarning yuqori darajadagi va murakkab abstrakt xususiyatlarini o'rganishda muhim rol o'ynaydi.

Konvolyutsion Qatlamlar (Convolutional Layers): Asosan tasvirni qayta ishlash sohasida (Computer Vision) ishlatiladi. Ular kiritilgan ma'lumotlarda joylashgan mahalliy xususiyatlarni (local features), masalan, chekkalarni yoki teksturalarni o'rganish uchun filtrlar yordamida konvolyutsiya operatsiyasini bajaradi.

Qaytariluvchi Qatlamlar (Recurrent Layers): Ketma-ket ma'lumotlar (Sequential Data), masalan, matn yoki vaqt qatorlari bilan ishlash uchun mo'ljallangan. Ular ma'lumotdagi ketma-ket bog'liqliklarni eslab qolishga qodir bo'lgan ichki holat (internal state) mexanizmlariga ega.

Neyron tarmoqni o'qitish jarayoni optimallashtiruvchilar va yo'qotish funksiyalari (Loss Functions) yordamida boshqariladi.

Yo'qotish Funksiyasi: Bu funksiya modelning bashoratlari va ma'lumotlarning haqiqiy qiymatlari o'rtasidagi farqni miqdoriy jihatdan baholaydi. U modelning qanchalik yomon ishlayotganini ko'rsatadi va modelni o'qitishda minimallashtirilishi kerak bo'lgan asosiy ko'rsatkichdir (masalan, Kvadratik Xatoliklar O'rtachasi regressiya muammolarida yoki Kross-entropiya tasniflash muammolarida).

Optimallashtiruvchilar: Bu algoritmlar yo'qotish funksiyasini minimallashtirish uchun modelning ichki parametrlarini (vaznlarini) iterativ ravishda sozlash vazifasini bajaradi. Ular gradiyentni hisoblash (gradient calculation) va vaznlarni yangilash tezligini (o'rganish tezligi) boshqarish kabi murakkab ishlarni amalga oshiradi (masalan, Adam, SGD yoki RMSprop kabi usullar).

Keras API'si barcha ushbu komponentlarni (qatlamlar, optimallashtiruvchilar va yo'qotish funksiyalari) bir joyga jamlab, tadqiqotchilarga chuqur o'rganish modellarini tezkor va tizimli ravishda qurish va tajriba o'tkazish uchun yagona va samarali interfeysni taqdim etadi.

Chuqur o'rganish modellarini muvaffaqiyatli o'qitishda modelning me'moriy tuzilishidan tashqari, ma'lumotlarni kiritish va qayta ishlashning samaradorligi ham fundamental ahamiyatga ega. Katta hajmli ma'lumotlar to'plamlari bilan ishlashda tez-tez uchraydigan asosiy muammo shundaki, markaziy protsessor (MPY) ma'lumotlarni tezkor saqlash moslamasidan (masalan, diskdan) yuklash, ularni qayta ishlash va grafik protsessorga (GPY) uzatish tezligi GPY ning hisoblash tezligiga mos kelmay qoladi. Bu holat GPYning bo'sh turib qolishiga olib keladi, bu esa o'qitish jarayonini sezilarli darajada sekinlashtiradi. Aynan shu muammoni hal etish uchun TensorFlow tf.data API ni taqdim etadi.

An'anaviy ma'lumotlarni yuklash usullari ko'pincha xotira cheklovlari va kiritish/chiqarish (I/O) tezligi muammolariga duch keladi. Agar ma'lumotlarning barchasi asosiy xotiraga (RAM) sig'masa, ularni har bir iteratsiyada diskdan yuklash zarur bo'ladi. Bu yuklash operatsiyalari MPY va GPY hisoblash operatsiyalariga nisbatan juda sekin bo'ladi. Samaradorlikni oshirish uchun ma'lumotlarni qayta ishlash va ularni GPYga uzatish jarayonlari hisoblashlardan parallel va asinxron ravishda amalga oshirilishi shart.

tf.data API ushbu muammolarni bartaraf etish uchun ma'lumotlarni yuklash va qayta ishlash uchun samarador ma'lumotlar quvurini (data pipeline) yaratish imkonini beradi. Bu quvur liniyasi o'zida bir qancha muhim transformatsiyalarni birlashtiradi:

1. Yuklash va Berish: Dastlab, ma'lumotlar manbadan (masalan, diskdagi fayllar, ma'lumotlar bazalari) yuklanadi va Dataset ob'ekti ko'rinishida taqdim etiladi.

2. Transformatsiya: Keyin, ma'lumotlarga zarur bo'lgan barcha oldindan qayta ishlash operatsiyalari (masalan, o'lchamini o'zgartirish, normalizatsiya) ketma-ket qo'llaniladi.

3. To'plamlash (Batching): Ma'lumotlar modelni o'qitish uchun mos bo'lgan kichik to'plamlarga (batches) guruhlanadi.

Ma'lumotlarni Oldindan Qayta Ishlash

tf.data quvur liniyasi, shuningdek, ma'lumotlarni oldindan qayta ishlash usullarini (preprocessing) integratsiya qilish uchun ideal joydir. Bunga misol sifatida normalizatsiya (kirish qiymatlarini standart diapazonga keltirish) va ma'lumotlarni oshirish (Data Augmentation) kabi usullarni keltirish mumkin. Ma'lumotlarni oshirish usuli (masalan, tasvirlarni tasodifiy burish yoki kattalashtirish) modelning umumlashtirish qobiliyatini oshirishga xizmat qiladi, bu esa uning test ma'lumotlaridagi aniqligini yaxshilaydi. tf.data

yordamida bu murakkab transformatsiyalar MPYda, GPY hisoblashlaridan parallel ravishda bajarilishi mumkin, bu esa butun o'qitish tsiklining samaradorligini kafolatlaydi.

O'qitilgan modelning sifatini baholash uchun faqatgina yo'qotish funksiyasi qiymatiga emas, balki aniq baholash metrikalariga (Evaluation Metrics) e'tibor qaratish kerak. Metrikalar modelning amaliy samaradorligini xolis va tushunarli tarzda aks ettiradi.

Tasniflash Muammolarida: Aniqlik (Accuracy), Pretsizion (Precision), Chaqiruv (Recall) va F-o'lchov (F-score) kabi metrikalar muhim hisoblanadi. Ular modelning to'g'ri va noto'g'ri tasniflangan holatlarni (True/False Positives/Negatives) qanday boshqarayotganini aniq ko'rsatadi.

Regressiya Muammolarida: Kvadratik Xatoliklar O'rtachasi (Mean Squared Error - MSE) yoki Mutlaq Xatoliklar O'rtachasi (Mean Absolute Error - MAE) kabi metrikalar model bashoratlari va haqiqiy qiymatlar orasidagi o'rtacha masofani o'lchaydi.

Modelni baholashda o'zaro tekshirish (cross-validation) usullari ham qo'llaniladi, bu modelning ma'lumotlardagi kutilmagan o'zgarishlarga qanchalik mustahkam (robust) ekanligini aniqlashga yordam beradi.

O'qitish va baholash jarayoni tugagach, modelni amaliyotda foydalanish uchun saqlash va joylashtirish zarur. TensorFlow buning uchun bir nechta standart formatlarni taklif qiladi:

SavedModel Formati: Bu TensorFlowning asosiy va tavsiya etilgan formatidir. U nafaqat modelning vaznlarini, balki butun hisoblash grafigini, optimallashtiruvchi holatini va barcha imzolangan funksiyalarni saqlaydi. Bu format modelni turli platformalar, shu jumladan boshqa dasturlash tillarida yozilgan ilovalar uchun ham universal tarzda eksport qilish imkonini beradi.

HDF5 Formati: Bu format asosan Keras modellarining vaznlarini saqlash uchun ishlatiladi, ammo ko'pincha qo'shimcha me'moriy ma'lumotlarni saqlash talab qilinadi.

Modelni Joylashtirish esa modelni real dunyo sharoitida foydalanuvchilar talablariga javob beradigan holatga keltirishni anglatadi. TensorFlow buning uchun keng qamrovli ekotizimni taqdim etadi:

TensorFlow Serving: Server muhitida yuqori unumdorlik bilan ishlash uchun mo'ljallangan, tarqatilgan tizim. U modelni bir nechta versiyalarda boshqarish va minimal kechikish bilan so'rovlarga javob berish imkonini beradi.

TensorFlow Lite: Mobil qurilmalar va o'rnatilgan tizimlar kabi resurslari cheklangan qurilmalar uchun optimallashtirilgan versiya. U modelni kichikroq hajmda va tezroq ijro etish uchun konvertatsiya qiladi.

TensorFlow.js: Modelni bevosita veb-brauzer muhitida yoki Node.js serverida ishga tushirish uchun JavaScript kutubxonasi.

Ushbu joylashtirish mexanizmlari TensorFlow modelining tadqiqot laboratoriyasidan tortib, iste'molchiga yo'naltirilgan mahsulotgacha bo'lgan to'liq hayot aylanishini yakunlashga imkon beradi.

**XULOSA**

Ushbu ilmiy tadqiqot ishida chuqur o'rganishning eng yetakchi freymvorklaridan biri bo'lgan TensorFlowning me'moriy, o'rnatish, funksional va amaliy jihatlari chuqur tahlil qilindi. Bizning tahlilimiz freymvorkning evolyutsiyasini statik graflar kontseptsiyasidan Eager Execution dinamik modeliga o'tish orqali kuzatib, uning zamonaviy chuqur o'rganish muammolarini hal qilishdagi moslashuvchanligini isbotladi.

**FOYDALANILGAN ADABIYOTLAR:**

1. Goodfellow I, Bengio Y, Courville A. Deep Learning. MIT Press, 2016.
2. Abadi M et al. TensorFlow: A System for Large-Scale Machine Learning. Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2016.
3. Chollet F. Keras: Deep Learning for Humans. (Asosiy Keras hujjatlari va kitob).
4. Yegulalp T. Mastering TensorFlow. Second Edition. O’Reilly Media, 2018.
5. TensorFlow rasmiy hujjatlari va API qo'llanmalari. (Google Developers).