PYTORCH FREYMVORKI: O'RNATISH VA ASOSIY FUNKSIYALAR

Tojimamatov Isroil Nurmamatovich
Farg‘ona davlat universiteti Amaliy matematika va informatika kafedrasi katta o’qituvchisi
E-mail: istailtojimamatov@gmail.com
Muhammadshokirova Dinora Ma'rufjon qizi
Farg‘ona davlat universiteti Amaliy matematika yo‘nalishi 3-bosqich 23.07-guruh talabasi
E-mail: dinorashermurodova91@gmail.com

Annotatsiya

Ushbu maqolada PyTorch freymvorkining texnik va nazariy asoslari, uning arxitekturasi, o‘rnatish jarayoni, muhitni sozlash bosqichlari hamda tensorlar, autograd tizimi, neyron tarmoqlar moduli, optimizatorlar va ma’lumotlar bilan ishlash mexanizmlari chuqur tahlil qilinadi. PyTorchning GPU bilan integratsiyasi, hisoblash grafigi, modullilik konsepsiyasi, yuqori samaradorlikdagi parallel hisoblash imkoniyatlari, amaliy misollar, kod fragmentlari, ilmiy tadqiqotlarda qo‘llanilishi, kompyuter ko‘rish, tabiiy tilni qayta ishlash, vaqt qatorlari tahlili va generativ modellar yaratishdagi afzalliklari har tomonlama yoritiladi. Maqola chuqur o‘rganish sohasiga oid ilmiy-texnik yondashuvlar va PyTorch freymvorkining zamonaviy sun’iy intellekt ekotizimidagi o‘rnini ko‘rsatishga qaratilgan.

Kalit so‘zlar: *PyTorch, Tensor, Deep Learning, Autograd, CUDA, GPU, Neyron tarmoq, Optimizator, DataLoader, AI Framework, kompyuter ko‘rish, NLP, Model, Backpropagation.*

Abstract. *This article thoroughly analyzes the technical and theoretical foundations of the PyTorch framework, its architecture, installation process, environment adjustment stages, as well as tensors, autograd system, neural network module, optimizers, and data processing mechanisms. The integration of PyTorch with a GPU, computational graphics, modularity concept, high-performance parallel computing capabilities, practical examples, code fragments, application in scientific research, computer vision, natural language processing, time series analysis, and the advantages of creating generative models are comprehensively highlighted. The article aims to demonstrate scientific and technical approaches in the field of deep learning and the role of the PyTorch framework in the modern artificial intelligence ecosystem.*

Keywords: *PyTorch, Tensor, Deep Learning, Autograd, CUDA, GPU, Neural network, Optimizer, DataLoader, AI Framework, computer vision, NLP, Model, Backpropagation.*

Аннотация. *В данной статье подробно анализируются технические и теоретические основы фреймворка PyTorch, его архитектура, процесс установки, этапы настройки среды, а также тензоры, автоградная система, модуль нейронных сетей, оптимизаторы и механизмы работы с данными. Всесторонне освещены интеграция PyTorch с GPU, вычислительный график, концепция модульности, возможности высокоэффективных параллельных вычислений, практические примеры, фрагменты кода, применение в научных исследованиях, компьютерное зрение, обработка естественного языка, анализ временных рядов и преимущества генеративных моделей. Статья направлена на демонстрацию научно-технических подходов в области глубокого изучения и роли фреймворка PyTorch в современной экосистеме искусственного интеллекта.*

Ключевые слова: *PyTorch, Tensor, Deep Learning, Autograd, CUDA, GPU, Neural network, Optimizator, DataLoader, AI Framework, computer vision, NLP, Model, Backpropagation.*

Kirish

XXI asrning eng muhim texnologiyalaridan biri — bu sun’iy intellekt (SI) hamda uning murakkab yo‘nalishi bo‘lgan chuqur o‘rganish (Deep Learning) hisoblanadi. Chuqur o‘rganish algoritmlarini samarali yaratish, sinash va optimallashtirish uchun maxsus kutubxonalar va freymvorklar talab etiladi. TensorFlow, Keras, Theano kabi freymvorklar orasida PyTorch o‘zining moslashuvchanligi, qulay sintaksisi va dinamik hisob grafigi bilan ajralib turadi.

PyTorch 2016-yilda Facebook AI Research (FAIR) tomonidan ishlab chiqilgan bo‘lib, bugungi kunda eng mashhur chuqur o‘rganish freymvorklaridan biri hisoblanadi. Uning mashhurlik darajasi akademik tadqiqotlardan tortib sanoat loyihalarigacha kengayib bormoqda. PyTorchning mashhurligining asosiy sababi — uning *dinamik computation graph, avtomatik differensiallash, GPU tezlashuvi, kuchli modul tizimi va tadqiqotlarga mos ochiq arxitekturasidir.*

PyTorchning asosiy ustunligi — uning moslashuvchanligi, intuitiv sintaksisi va GPU resurslaridan samarali foydalanish imkoniyatidir. PyTorch arxitekturasi bir nechta asosiy komponentlardan tashkil topgan. ***Torch.Tensor*** — freymvorkning markaziy obyektidir. Bu ko‘p o‘lchamli massiv bo‘lib, u CPU yoki GPUda saqlanishi va hisob-kitoblarga jalb qilinishi mumkin. Tensorlar yordamida vektor, matritsa va yuqori o‘lchovli tuzilmalar ustida amallar bajariladi. ***Autograd*** moduli esa avtomatik ravishda gradientlarni hisoblaydi. Bu mexanizm neyron tarmoqlarning orqaga tarqatish (backpropagation) jarayonini avtomatlashtirib, model o‘qitilishini sezilarli darajada soddalashtiradi. Neyron tarmoqlarni yaratish uchun ***torch.nn*** moduli mo‘ljallangan. Unda qatlamlar (Linear, Conv2d, LSTM va boshqalar), faollashtiruvchi funksiyalar va

yo‘qotish funksiyalari jamlangan. Parametrlarni yangilash uchun ***torch.optim*** moduli qo‘llaniladi; u ichiga SGD, Adam, RMSprop kabi mashhur optimizatsiya algoritmlarini oladi. Ma’lumotlar bilan ishlash jarayonida ***Dataset*** va ***DataLoader*** obyektlari katta hajmdagi ma’lumotlarni yuklash, bo‘lish va batching qilishni avtomatlashtiradi. PyTorchning ommalashishiga sabab — uning Pythonic sintaksisi, ilmiy tadqiqotlarda qulayligi va GPU bilan mukammal integratsiyasidir. Bugungi kunda u akademik tadqiqotlardan tortib sanoatdagi real AI tizimlarigacha keng qo‘llanilmoqda.

PyTorch freymvorkidan samarali foydalanish uchun avvalo uning to‘g‘ri o‘rnatilishi va tizim talablariga mos kelishi muhimdir. PyTorch Windows, Linux va macOS operatsion tizimlarida erkin ishlaydi. Dasturiy ta’minotning barqaror ishlashi uchun *Pythonning 3.8–3.12 versiya*lari tavsiya etiladi. Agar foydalanuvchi NVIDIA GPU asosida tezlashtirilgan hisob-kitoblardan foydalanmoqchi bo‘lsa, *CUDA 11.8* yoki *12.x* versiyalaridan biri tizimda mavjud bo‘lishi lozim. GPU bo‘lmagan holatlarda esa PyTorchning CPU versiyasi yetarli bo‘ladi.

PyTorchni o‘rnatish jarayoni juda sodda. CPU versiyasi uchun *pip install torch torchvision torchaudio* buyrug‘i kifoya qiladi. Agar foydalanuvchi CUDA bilan GPU versiyasidan foydalanmoqchi bo‘lsa, maxsus indeks manzili yordamida *pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118* buyrug‘i bajariladi. O‘rnatilganini tekshirish uchun *torch.__version__* funksiyasi versiyani ko‘rsatadi, *torch.cuda.is_available()* esa GPU mavjud yoki yo‘qligini aniqlaydi. Agar GPU aniqlansa, *torch.cuda.get_device_name(0)* orqali grafik protsessor nomi ham chiqadi. PyTorchning eng asosiy elementi — bu **tensor**. Tensor — ko‘p o‘lchamli massiv bo‘lib, tuzilishi jihatidan NumPy massivlariga o‘xshaydi, biroq eng katta afzalligi GPUda ishlay olishidir. Tensorlar *torch.tensor(), torch.zeros(), torch.randn()* kabi funksiyalar orqali yaratiladi. Har bir tensorning o‘lchami *(shape)*, ma’lumot turi *(dtype)* va qaysi qurilmada saqlanayotgani *(device)* tekshirilishi mumkin. Tensorlarni GPUga yuklash uchun to('cuda') metodi qo‘llaniladi. Tensorlar ustida qo‘shish, ko‘paytirish, matritsalar bilan ishlash, eksponenta, trigonometrik funksiyalar kabi element-wise amallar bajarilishi mumkin. Shuningdek, PyTorch va NumPy o‘rtasida oson integratsiya mavjud bo‘lib, tensorlarni .numpy() orqali NumPy massiviga va *torch.from_numpy()* orqali tensor formatiga o‘tkazish mumkin. PyTorchning kuchli tomonlaridan biri ham ana shu moslashuvchanlikdir.

PyTorchning eng muhim komponentlaridan biri — bu Autograd tizimi bo‘lib, u avtomatik differensiallash mexanizmini ta’minlaydi. Autograd yordamida model parametrlari uchun gradientlar avtomatik hisoblanadi va bu, ayniqsa, neyron tarmoqlarni o‘qitish jarayonida orqa tarqalish (backpropagation) algoritmini sodda va qulay shaklda amalga oshirish imkonini beradi. Har qanday tensorni gradient hisoblash jarayoniga jalb qilish uchun *requires_grad=True* bayrog‘i qo‘yiladi. Oddiy misolda, *x*

= torch.tensor(3.0, requires_grad=True) tarzida yaratilgan o‘zgaruvchi ustida y = x**3 + 2*x funksiyasi hisoblanadi. *y.backward()* chaqirilgach, PyTorch hosilani avtomatik ravishda hisoblab, *x.grad* ga saqlaydi. Bir nechta o‘zgaruvchilar bilan ishlashda ham Autograd mustaqil ravishda barcha gradientlarni topadi. Har bir iteratsiyada gradientlarni qayta to‘g‘ri hisoblash uchun *a.grad.zero_()* orqali gradientlar tozalanadi.

PyTorchda neyron tarmoqlarni yaratish *torch.nn* moduliga asoslanadi. Tarmoqlar odatda *nn.Module* klassidan meros olish orqali quriladi. Oddiy model misolida ikkita chiziqli qatlam *(Linear)* va *ReLU* faollashtirish funksiyasi qo‘llanadi. Forward-pass jarayonida ma’lumotlar birinchi chiziqli qatlamga uzatiladi, ReLU orqali faollashtiriladi va yakuniy qatlamdan o‘tkaziladi. PyTorchning ushbu moduli ReLU, Sigmoid, Tanh, LeakyReLU, Softmax kabi ko‘plab faollashtirish funksiyalarini taqdim etadi. Modelni o‘qitish jarayonida optimallashtirish algoritmlari muhim o‘rin tutadi. PyTorchdagi *torch.optim* moduli SGD, Adam, RMSprop, Adagrad va Adadelta kabi mashhur optimizatorlarni o‘z ichiga oladi. O‘qitish siklida *optimizer.zero_grad()* orqali gradientlar tozalanadi, modeldan o‘tgan ma’lumotlar yordamida loss hisoblanadi, so‘ng *loss.backward()* bilan gradientlar topiladi va *optimizer.step()* orqali parametrlar yangilanadi.

Ma’lumotlar bilan ishlashda esa Dataset va DataLoader modullari qo‘llaniladi. Dataset klassi foydalanuvchi tomonidan maxsus yozilib, unda ma’lumotlarning uzunligi va indeks bo‘yicha elementlarni qaytarish funksiyalari aniqlanadi. DataLoader esa ushbu ma’lumotlarni batchlarga ajratib, modelga qulay tarzda uzatishni ta’minlaydi. Bu PyTorchni katta hajmdagi ma’lumotlar bilan samarali ishlash imkonini beradigan qudratli freymvorkga aylantiradi.

PyTorchning amaliy qo‘llanilish yo‘nalishlari juda keng bo‘lib, eng ko‘p qo‘llaniladigan asosiy sohalar quyidagilardan iborat. Birinchi yirik yo‘nalish — ***kompyuter ko‘rish (Computer Vision)***. Bu sohada tasvirlarni aniqlash, tasniflash, obyektlarni segmentatsiya qilish va ularni real vaqt rejimida kuzatish kabi murakkab vazifalar bajariladi. PyTorchda ResNet, VGG, AlexNet kabi chuqur konvolyutsion neyron tarmoqlar (CNN) keng qo‘llaniladi. Bundan tashqari, ob’ekt aniqlash uchun YOLO, Faster R-CNN kabi zamonaviy modellar PyTorch asosida yaratilgan va rivojlantirilmoqda. Tibbiyot tasvirlari segmentatsiyasi, geografik xaritalarni qayta ishlash va boshqa sohalarda esa U-Net modeli juda yuqori aniqlik beradi. PyTorchning moslashuvchanligi ushbu modellarni oson modifikatsiya qilish, sinovdan o‘tkazish va real vaqt tizimlariga joriy etish imkonini yaratadi.

Ikkinchi muhim yo‘nalish — ***tabiiy tilni qayta ishlash (NLP)***. Hozirgi kunda Transformer arxitekturasi NLP bo‘yicha universal standartga aylangan bo‘lib, BERT, GPT va RoBERTa kabi eng kuchli til modellarining barchasi PyTorch asosida ishlab chiqilgan. Seq2Seq modellar esa tarjima, matn yaratish va dialog tizimlarida samarali

qo‘llanadi. PyTorchning tensor va autograd tizimi matnlardagi murakkab kontekstual bog‘lanishlarni chuqur o‘rganish imkonini beradi. Shuning uchun ilmiy jamoalar va sanoat korxonalari NLP modellarining jahon miqyosidagi rivojlanishida PyTorchni asosiy vosita sifatida tanlashmoqda.

Uchinchi yo‘nalish — ***generativ modellar***. GANlar (Generative Adversarial Networks) yordamida realga yaqin tasvirlar generatsiyasi, VAE (Variational Autoencoder) orqali ma’lumotlar latent fazoda ifodalanishi, diffusion modellari orqali yuqori sifatli tasvirlar generatsiyasi keng qo‘llanadi. Ayniqsa, so‘nggi yillarda diffusion modellari (Stability AI, Midjourney, OpenAI tizimlari) PyTorch asosida ishlab chiqilayotgani sababli ushbu freymvork generativ sun’iy intellektning asosiy ekotizimiga aylangan.

To‘rtinchi yo‘nalish — ***mustahkamlovchi o‘qitish (Reinforcement Learning)***. PyTorch Stable-Baselines3 va RLlib kabi RL-kutubxonalar bilan to‘liq mos bo‘lib, robototexnika, o‘yinlar, optimizatsiya tizimlari va avtonom boshqaruv kabi sohalarda qo‘llaniladi. Dinamik grafikning mavjudligi agentlarning muhit bilan real vaqtda o‘zaro ta’sirini modellashtirishda ayniqsa qulaylik yaratadi.

PyTorchning ommabopligiga sabab bo‘lib xizmat qiluvchi asosiy afzalliklari — uning dinamik grafigi, GPU orqali hisoblashlarni bir necha baravar tezlashtirish imkoniyati, pythonic va oson o‘rganiladigan sintaksisi, katta global hamjamiyati va ilmiy tadqiqotlarga moslashgan qulay arxitekturasi hisoblanadi. Shu sababli PyTorch bugungi kunda sun’iy intellektning deyarli barcha asosiy yo‘nalishlarida yetakchi texnologiya sifatida qo‘llanilmoqda.

Mashinasozlik, kimyo texnologiyalari, kompyuter ko‘rish, tabiiy tilni qayta ishlash va boshqa ko‘plab sohalarda chuqur o‘rganish modellari tobora keng qo‘llanilmoqda. PyTorch freymvorki murakkab neyron tarmoqlarni qurish va ularni samarali o‘qitish uchun juda qulay imkoniyatlarga ega. Ushbu bo‘limda biz PyTorch yordamida ***bosqichma-bosqich murakkab regressiya modelini yaratish***, *unga* ***ma’lumotlar berish, model arxitekturasini belgilash*** *va* ***trening jarayonini amalga oshirish*** kabi amaliy jarayonlarni ko‘rib chiqamiz. Misolda regressiya vazifasi uchun to‘liq kod keltiriladi: avval sintetik dataset generatsiya qilinadi, so‘ng uch qatlamli neyron tarmoq yaratiladi va oxirida to‘liq trening sikli bajariladi.

Dataset tayyorlash

Modelni o‘qitishdan oldin unga mos ma’lumotlar to‘plami kerak bo‘ladi. Masalan, 500 ta kuzatuvdan iborat 10 o‘lchamli xususiyatlar (features) va ularga mos 1 ta chiqish (target) qiymati generatsiya qilinadi. Bunday ma’lumotlar ko‘pincha real sanoat jarayonlarining matematik modellashtirilgan versiyasini sinovdan o‘tkazish uchun ishlatiladi. PyTorch yordamida random ma’lumot generatsiya qilish juda oson:

```
X = torch.randn(500, 10)
Y = torch.randn(500, 1)
```

Bu yerda X — 500×10 o‘lchamli input matritsa, Y esa mos 500×1 o‘lchamli target vektordir. Bunday ma’lumotlar regressiya modelini sinovdan o‘tkazish uchun yetarli.

Model arxitekturasi

Model murakkabligi, qatlamlar soni va aktivatsiya funksiyalari uning o‘rganish qobiliyatiga ta’sir qiladi. Quyidagi sinf (class) yordamida ***uch yashirin qatlamli regressiya tarmog‘i*** yaratiladi:

```
class RegressionNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(10, 32)
        self.fc2 = nn.Linear(32, 16)
        self.fc3 = nn.Linear(16, 1)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        return self.fc3(x)
```

Modelning ishlash prinsipi:

fc1: 10 → 32 o‘tish, ya’ni kirishdagi 10 ta xususiyat 32 o‘lchamli yashirin qatlamga o‘tkaziladi.

fc2: 32 → 16 o‘tish, tarmoq diniylashtiriladi.

fc3: yakuniy 16 → 1 o‘tish, natijada model bitta regressiya qiymatini beradi.

ReLU aktivatsiya funksiyasi nolinearlik kiritadi va modelning moslashuvchanligini oshiradi.

Bunday model ko‘pincha kimyoviy jarayonlar matematikasi, absorbsiya ustunlaridagi massa almashinish jarayonlarini bashoratlash, gidravlik bosimni hisoblash, hamda ishlab chiqarishdagi optimallashtirish vazifalarida qo‘llanadi.

Trening sikli

Modelni o‘qitish uchun quyidagi elementlar kerak bo‘ladi:

Loss function (yo‘qotish funksiyasi) — MSE (Mean Squared Error) regressiya uchun eng mashhur mezon.

Optimizer (optimizator) — Adam algoritmi gradientlarni yangilashda juda barqaror va tez.

Modelni o‘qitish uchun to‘liq sikl:

```
model = RegressionNet()
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

```
for epoch in range(500):
    optimizer.zero_grad()
    y_pred = model(X)
    loss = criterion(y_pred, Y)
    loss.backward()
    optimizer.step()
```

Trening jarayonidagi asosiy bosqichlar:

1. ***zero_grad()*** — avvalgi siklдagi gradientlarni tozalaydi.
2. ***model(X)*** — modelga input beriladi va output olinadi.
3. ***loss = criterion(y_pred, Y)*** — bashorat bilan haqiqiy qiymatlar orasidagi farq o‘lchanadi.
4. ***loss.backward()*** — tarmoqdagi barcha parametrlarga nisbatan gradientlar hisoblanadi.
5. ***optimizer.step()*** — gradientlarga asoslanib tarmoq vaznlari yangilanadi.

500 epoch o‘qitilgan model anchagina moslashgan va real regressiya vazifalarini bajarishga tayyor bo‘ladi.

Xulosa

PyTorch bugungi kunda chuqur o‘rganish va sun’iy intellekt sohasida eng keng qo‘llaniladigan, qulay va kuchli freymvorklardan biridir. Uning o‘rnatish jarayoni sodda bo‘lib, Python muhiti va GPU drayverlari bilan yuqori darajada moslashadi. Dinamik hisoblash grafigi PyTorchning asosiy afzalliklaridan biri bo‘lib, u modelni real vaqt rejimida o‘zgartirish, test qilish va murakkab arxitekturalar bilan ishlash imkonini beradi. Freymvorkning Tensor tuzilmasi, avtomatik differensiallash (autograd) tizimi, neyron tarmoqlar uchun keng qamrovli torch.nn moduli va optimallashtirish uchun torch.optim paketi uni amaliyotda juda qulay qiladi. Bundan tashqari, PyTorch laboratoriya tajribalaridan tortib sanoat miqyosidagi modellarni ishlab chiqishgacha bo‘lgan jarayonlarda o‘zining barqarorligi va moslashuvchanligini namoyon etadi. Kompyuter ko‘rish, NLP, generativ modellar, vaqt qatorlari tahlili va Reinforcement Learning kabi ko‘plab sohalarda PyTorchning kuchli ekotizimi mavjud. Umuman olganda, PyTorchning soddaligi, tezligi, intuitiv sintaksisi va keng hamjamiyati uni zamonaviy AI tadqiqotlari va ishlab chiqish jarayonlarida eng ishonchli vositalardan biriga aylantiradi.

Foydalanilgan adabiyotlar

1. Bekmuratov Q.A. , Sun’iy intellekt va neyron tarmoqlar. O‘quv qo‘llanma, Samarqand – 2021.
2. Sadullayeva SH.A, Yusupov D.F., Yusupov F., Sun’iy intellect va neyronto‘rli texnologiyalar. O‘quv qo‘llanma, Urganch – 2021.
3. H.N.Zayniddinov, T.A.Xo‘jaqulov, M.P.Atadjanov, “Sun’iy intellekt” fanidan o‘quv qo‘llanma, Toshkent – 2018.

4. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Advances in Neural Information Processing Systems, 32, 8024-8035.
5. Official PyTorch Documentation. Available at: https://pytorch.org/docs/stable/index.html (Accessed: 2025).
6. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. [Online] Available at: https://www.deeplearningbook.org/
7. Интеллектуальные информационные системы и технологии: учебное пособие/ Ю.Ю. Громов, О.Г.Иванова, В.В. Алекссев и др. –тамбов: Издво ФГБОУ ВПО «ТГТУ», 2013.-244с.
8. Поталов А.С. Технологии искусственного интеллекта-СПб: СПбГУ ИТМОб 2010-218 с.

Elektron manbalar:

1. Official PyTorch Documentation. Available at: https://pytorch.org/docs/stable/index.html (Accessed: 2025).
2. www.lex.uz - O‘zbekiston Respublikasi Qonun hujjatlari milliy bazasi
3. htpp://www.raii.org/library - Российская ассоциация искусственного интеллекта
4. https://www.javatpoint.com/artificial-intelligence-ai