АО «АЛМАТИНСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет инженерных и информационных технологий

Кафедра компьютерной инженерии

ПРОЕКТИРОВАНИЕ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

ЛЕКЦИЯ № 7

Проектирование систем глубокого обучения

Теоретико-практический конспект

Составитель: Алтыбай А.

Кафедра компьютерной инженерии

Алматы, 2026

Аннотация и назначение лекции

Лекция 7 раскрывает тему «Проектирование систем глубокого обучения» в контексте разработки промышленной AI-системы. Материал опирается на тематический план дисциплины и соединяет определения, принципы проектирования, математические оценки, проектные артефакты, практический код и аргументированный анализ рисков. Вопросы построены вокруг решений, которые предстоит принимать инженеру, а не только вокруг названий алгоритмов. [1, 3]

План занятия

1. DL как компонент системы
2. Архитектуры по модальности
3. Обучение на GPU
4. Функция потерь и данные
5. Цикл обучения
6. Числовая устойчивость
7. Регуляризация
8. Transfer learning
9. Практические сценарии, контроль качества и проектные задания.

Ожидаемые результаты обучения

По завершении студент способен:

1. объяснить и аргументированно применить принцип «DL как компонент системы» к инженерному решению;
2. объяснить и аргументированно применить принцип «Архитектуры по модальности» к инженерному решению;

3. объяснить и аргументированно применить принцип «Обучение на GPU» к инженерному решению;
4. объяснить и аргументированно применить принцип «Функция потерь и данные» к инженерному решению;
5. объяснить и аргументированно применить принцип «Цикл обучения» к инженерному решению;
6. объяснить и аргументированно применить принцип «Числовая устойчивость» к инженерному решению;
7. реализовать упрощённый проверяемый прототип и сформулировать критерии его приемки.
8. указать угрозы достоверности результатов и предложить проверки до эксплуатации.

Сквозной кейс. В течение лекции рассматриваем визуальный контроль продукции. Существенные условия: переобучение сети, ошибочный preprocess. Основная проверяемая характеристика: recall дефектов и время обработки. Числа в расчётных примерах учебные и не являются результатами реальных измерений.

1. Базовые теоретические положения из конспекта курса

Ниже представлены исходные положения учебно-методического конспекта, связанные с данной темой. Следующие разделы раскрывают их с инженерными деталями и проверяемыми примерами. [1, 3] Цель лекции. показать инженерные особенности глубокого обучения: выбор архитектуры, GPU-ресурсы, transfer learning, управление экспериментами и воспроизводимость.

Ключевые понятия. deep learning, GPU, batch size, transfer learning, fine-tuning, checkpoint, seed, mixed precision, reproducibility, experiment tracking.

1.1 Почему DL требует отдельного проектирования

Глубокие сети часто имеют миллионы или миллиарды параметров, требуют GPU/accelerator и длительного обучения. Поэтому стоимость ошибки в эксперименте выше. Нужно планировать memory budget, batch size, checkpoints, контроль learning curves и повторное использование pretrained weights.

Инженерная задача включает не только архитектуру сети, но и загрузку данных, эффективность GPU utilization, хранение артефактов и механизм возобновления обучения.

1.2 Transfer learning и fine-tuning

Transfer learning использует знания pretrained модели. Часто сначала замораживают основную часть сети и обучают небольшой head, затем при необходимости размораживают часть слоев и выполняют fine-tuning с меньшим learning rate.

Это сокращает требования к данным и вычислениям, но требует учитывать доменный сдвиг и лицензионные/эксплуатационные ограничения исходной модели.

1.3 Воспроизводимость эксперимента

Для воспроизводимости фиксируют код, dataset version, конфигурацию модели, seed, версии библиотек и аппаратную среду. Однако на GPU абсолютная детерминированность может быть дорогой или недоступной; поэтому важно различать повторяемость эксперимента и статистическую устойчивость результатов.

Хорошая практика — сохранять конфигурацию эксперимента рядом с checkpoint и метриками.

1.4 Управление ресурсами

Batch size влияет на память и динамику оптимизации. Mixed precision ускоряет вычисления и уменьшает память, но требует контроля численной устойчивости. Gradient accumulation позволяет имитировать большой batch при ограниченной памяти. Distributed training используется, когда одна карта недостаточна.

Перед масштабированием полезно профилировать узкие места: иногда GPU простаивает из-за медленной загрузки данных.

1.5 Минимальный набор артефактов DL-эксперимента

Рекомендуется сохранять: конфигурацию, commit кода, dataset identifier, checkpoint, optimizer state, learning curves, метрики на validation, пример входа и правила preprocessing. Эти артефакты позволяют воспроизвести обучение и корректно развернуть модель.

пример фиксации seed в PyTorch import random, numpy as np, torch seed = 42 random.seed(seed); np.random.seed(seed) torch.manual_seed(seed) if torch.cuda.is_available(): torch.cuda.manual_seed_all(seed)

1.6 Выводы

▷ DL-проект требует управления вычислениями и артефактами, а не только выбора сети.

▷ Transfer learning снижает стоимость обучения, но не устраняет необходимость валидации на своем домене.

▷ Воспроизводимость требует версий кода, данных, конфигурации и среды.

▷ Профилирование ресурсов должно предшествовать дорогостоящему масштабированию.

1.7 Контрольные вопросы

- ▷ Что нужно сохранить для воспроизводимости DL-эксперимента?
- ▷ В чем отличие transfer learning и fine-tuning?
- ▷ Как batch size связан с памятью GPU?
- ▷ Для чего используется mixed precision?
- ▷ Почему одинаковый seed не всегда гарантирует абсолютно идентичный результат на GPU?

2. Систематическое изложение темы и инженерные решения

2.1 DL как компонент системы

Нейронная сеть выполняет функцию в более широком pipeline: обучение, checkpoint, загрузка, inference и мониторинг. Самостоятельный torch.nn.Module не образует законченный AI-сервис. [2]

> **Проверяемый пример.** Спроектируйте вход, тензорные преобразования, сеть, постобработку и API для изображения.

Вопрос для обсуждения. Как проверить выполнение такого требования: Спроектируйте вход, тензорные преобразования, сеть, постобработку и API для изображения.

2.2 Архитектуры по модальности

MLP подходит для базовых dense-признаков, CNN для локальных паттернов изображений, Transformer для последовательностей и токенов. Выбор зависит от инвариантностей данных. [3]

Проверяемый пример. Сравните трудоемкость MLP и CNN на изображении 224x224.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сравните трудоемкость MLP и CNN на изображении 224x224.

2.3 Обучение на GPU

Перенос данных на accelerator полезен при достаточной вычислительной нагрузке; небольшие задачи могут работать быстрее на CPU. Измерять нужно полный end-to-end pipeline. [4]

Проверяемый пример. Проверьте, что модель и batch находятся на одном устройстве, включая labels.

Вопрос для обсуждения. Как проверить выполнение такого требования: Проверьте, что модель и batch находятся на одном устройстве, включая labels.

2.4 Функция потерь и данные

Размерности logits и целевых меток должны соответствовать постановке; CrossEntropyLoss ожидает logits, а не уже примененный softmax. [5]

Проверяемый пример. Для 3 классов определите формы tensor logits (N,3) и target (N,).

Вопрос для обсуждения. Как проверить выполнение такого требования: Для 3 классов определите формы tensor logits (N,3) и target (N,).

2.5 Цикл обучения

Train mode, zero_grad, forward, loss, backward, optimizer step образуют цикл обновления. Eval mode и отсутствие записи градиентов важны при проверке. [6]

Проверяемый пример. Объясните роль model.eval() для Dropout и BatchNorm.

Вопрос для обсуждения. Как проверить выполнение такого требования: Объясните роль model.eval() для Dropout и BatchNorm.

2.6 Числовая устойчивость

Взрывающиеся градиенты, переполнение softmax и неподходящий learning rate приводят к NaN. Стандартные функции потерь и clipping помогают стабилизации. [1]

Проверяемый пример. Добавьте проверку finiteness loss и градиентов до сохранения checkpoint.

Вопрос для обсуждения. Как проверить выполнение такого требования: Добавьте проверку finiteness loss и градиентов до сохранения checkpoint.

2.7 Регуляризация

Dropout, weight decay, data augmentation и early stopping по-разному изменяют обобщение модели. Настройка требует независимого validation. [2]

Проверяемый пример. Исследуйте gap между training и validation loss для пяти эпох.

Вопрос для обсуждения. Как проверить выполнение такого требования: Исследуйте gap между training и validation loss для пяти эпох.

2.8 Transfer learning

Предобученная модель позволяет переиспользовать представления, но требует адаптации входных преобразований и выходного слоя. Заморозка параметров меняет число обучаемых весов. [3]

Проверяемый пример. Замените финальный классификатор ResNet18 и обсудите fine-tuning с различными learning rates.

Вопрос для обсуждения. Как проверить выполнение такого требования: Замените финальный классификатор ResNet18 и обсудите fine-tuning с различными learning rates.

2.9 Checkpointing

Сохраняют веса, метрики, архитектуру, конфигурацию и иногда состояние optimizer. Best checkpoint выбирают по validation, а не по training loss. [4]

Проверяемый пример. Сравните модель последней эпохи и лучшую по val F1.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сравните модель последней эпохи и лучшую по val F1.

2.10 Репродуцируемость и версии

Указывают seed, версии torch, device, доступные операции и набор данных. В случае nondeterministic kernels отмечают источник различий. [5]

Проверяемый пример. Составьте manifest с data_hash, commit, torch_version и best_epoch.

Вопрос для обсуждения. Как проверить выполнение такого требования: Составьте manifest с data_hash, commit, torch_version и best_epoch.

2.11 Инференс и вычислительные ограничения

Batching и precision могут повысить throughput, но влияют на память, задержки и численную точность. Выбор режима измеряется на целевом оборудовании. [6]

Проверяемый пример. Сравните batch size 1 и 32 по latency на объект и общему throughput.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сравните batch size 1 и 32 по latency на объект и общему throughput.

2.12 Документирование DL-модели

Model card сообщает назначение, данные, риск, ограничения и протокол теста. Оно должно обновляться вместе с версией весов. [1]

Проверяемый пример. Заполните раздел ограничения по плохому освещению для классификатора изображений.

Вопрос для обсуждения. Как проверить выполнение такого требования: Заполните раздел ограничения по плохому освещению для классификатора изображений.

3. Математические инструменты и количественные критерии

Для перехода от неформального решения к проверяемой постановке введём следующие величины.

$$z^{(\ell)} = W^{(\ell)} a^{(\ell-1)} + b^{(\ell)}, \qquad a^{(\ell)} = \sigma(z^{(\ell)}) \tag{1}$$

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t) \tag{2}$$

В прямом проходе формируются предсказания, в обратном — градиенты. При оценке модели параметры не меняются.

При выборе числового критерия нужно отличать обучающую функцию потерь, продуктовую ценность и эксплуатационные ограничения. Первую можно оптимизировать дифференцированием, вторая определяется процессом принятия решения, а третьи проверяются нагрузочными испытаниями, политиками безопасности и мониторингом.

3.1 Пошаговый расчёт на учебных данных

Рассмотрим две конфигурации одной системы. Вариант A имеет условную оценку качества $q_A = 0.86$ и задержку $t_A = 90$ мс; вариант B — $q_B = 0.89$ и $t_B = 280$ мс. Предположим, что требование заказчика задаёт $q \geq 0.85$ и $p95 \leq 150$ мс. Тогда A удовлетворяет обоим условиям, тогда как B нарушает второе. Следовательно, выбор B только потому, что его offline-метрика выше, не является обоснованным. Числа иллюстративны; реальный вывод требует измерений при одинаковой нагрузке и одинаковом наборе данных.

$$\begin{aligned} \text{accept}(A) &= \mathbf{1}\{0.86 \geq 0.85 \wedge 90 \leq 150\} = 1, \\ \text{accept}(B) &= \mathbf{1}\{0.89 \geq 0.85 \wedge 280 \leq 150\} = 0. \end{aligned}$$

3.2 Как интерпретировать число, а не только вычислить его

Сначала следует проверить единицы измерения, состав выборки и смысл агрегирования. Например, p95 на спокойном стенде нельзя переносить на рабочую нагрузку с иным числом одновременных запросов. Затем исследуют чувствительность к изменениям входных данных и неопределённость оценок. Наконец, фиксируют последствия порога: какую ошибку он предотвращает, а какую может увеличить. Этот подход делает количественный критерий частью инженерной аргументации, а не украшением отчёта. [1, 3]

4. Практическое программирование на Python

Пример предназначен для демонстрации ключевого инженерного принципа. Он не претендует на обработку реального production-трафика. Запускать код следует в изолированном окружении и фиксировать версии библиотек, если они используются. Вывод зависит от указанных входных данных; для стохастических алгоритмов он может меняться.

4.1 Основной воспроизводимый пример

```
import torch
from torch import nn

torch.manual_seed(17)
net = nn.Sequential(nn.Linear(8, 16), nn.ReLU(),
                    nn.Dropout(0.1), nn.Linear(16, 3))
optim = torch.optim.Adam(net.parameters(), lr=1e-3)
features = torch.randn(32, 8)
target = torch.randint(0, 3, (32,))
net.train()
optim.zero_grad()
logits = net(features)
loss = nn.CrossEntropyLoss()(logits, target)
loss.backward()
optim.step()
net.eval()
with torch.inference_mode():
    print("prediction shape:", tuple(net(features).shape))
```

Проверка результатов. Сначала убедитесь, что код завершился без исключений; проверьте формы массивов, схему результата и отсутствие недопустимых значений. Если используются внешние данные, включите в отчёт способ их получения и идентификатор версии. Измените один входной параметр и объясните, почему наблюдаемый результат изменился или остался прежним.

4.2 Компактная проверка контракта или метрики

```python
total = sum(p.numel() for p in net.parameters())
trainable = sum(p.numel() for p in net.parameters() if p.requires_grad)
print("parameters:", total, "trainable:", trainable)
```

При самостоятельном повторении предложите минимум один негативный тест: некорректное значение, пустой вход, несовместимую версию схемы либо нарушение SLO. Переход от успешного демонстрационного запуска к надежной системе требует именно таких проверок.

5. Разбор трёх инженерных ситуаций

5.1 Ситуация 1: DL как компонент системы

Заказчик использует визуальный контроль продукции. Новое требование заставляет рассмотреть принцип «DL как компонент системы». Основная неопределённость состоит в том, как одновременно обеспечить recall дефектов и время обработки, не допустить ошибочный preprocess и сохранить возможность изменения модели.

Постановка. Нейронная сеть выполняет функцию в более широком pipeline: обучение, checkpoint, загрузка, inference и мониторинг. Самостоятельный torch.nn.Module не образует законченный AI-сервис.

Действие инженера. Спроектируйте вход, тензорные преобразования, сеть, постобработку и API для изображения.

Альтернативы. Сравните три подхода: (A) простое правило без ML, (B) модульное ML-решение с отдельной проверкой входных данных и (C) интегрированный сервис со сложной моделью. Для каждого укажите потребность в данных, ожидаемые инженерные риски, возможность тестирования и минимальную инфраструктуру. Таблица ниже задаёт форму отчёта, а не готовый рейтинг.

Вариант	Данные и зависимости	Проверка	Основной риск
Правила	Фиксированные условия	Набор тест-кейсов	Ограниченная гибкость
ML-модуль	Версионированные признаки	Offline и API tests	Drift и skew
Сложный сервис	Несколько компонентов	End-to-end и нагрузка	Каскадный отказ

Проверка решения. Подготовьте две позитивные и две негативные пробы; задайте метрику успеха, базовый сценарий и критерий остановки внедрения. Обсудите, какие допущения останутся непроверенными даже после удачного теста.

5.2 Ситуация 2: Числовая устойчивость

Заказчик использует визуальный контроль продукции. Новое требование заставляет рассмотреть принцип «Числовая устойчивость». Основная неопределённость состоит в том, как одновременно обеспечить recall дефектов и время обработки, не допустить ошибочный preprocess и сохранить возможность изменения модели.

Постановка. Взрывающиеся градиенты, переполнение softmax и неподходящий learning rate приводят к NaN. Стандартные функции потерь и clipping помогают стабилизации.

Действие инженера. Добавьте проверку finiteness loss и градиентов до сохранения checkpoint.

Альтернативы. Сравните три подхода: (A) простое правило без ML, (B) модульное ML-решение с отдельной проверкой входных данных и (C) интегрированный сервис со сложной моделью. Для каждого укажите потребность в данных, ожидаемые инженерные риски, возможность тестирования и минимальную инфраструктуру. Таблица ниже задаёт форму отчёта, а не готовый рейтинг.

Вариант	Данные и зависимости	Проверка	Основной риск
Правила	Фиксированные условия	Набор тест-кейсов	Ограниченная гибкость
ML-модуль	Версионированные признаки	Offline и API tests	Drift и skew
Сложный сервис	Несколько компонентов	End-to-end и нагрузка	Каскадный отказ

Проверка решения. Подготовьте две позитивные и две негативные пробы; задайте метрику успеха, базовый сценарий и критерий остановки внедрения. Обсудите, какие допущения останутся непроверенными даже после удачного теста.

5.3 Ситуация 3: Инференс и вычислительные ограничения

Заказчик использует визуальный контроль продукции. Новое требование заставляет рассмотреть принцип «Инференс и вычислительные ограничения». Основная неопределённость состоит в том, как одновременно обеспечить recall дефектов и время обработки, не допустить ошибочный preprocess и сохранить возможность изменения модели.

Постановка. Batching и precision могут повысить throughput, но влияют на память, задержки и численную точность. Выбор режима измеряется на целевом оборудовании.

Действие инженера. Сравните batch size 1 и 32 по latency на объект и общему throughput.

Альтернативы. Сравните три подхода: (A) простое правило без ML, (B) модульное ML-решение с отдельной проверкой входных данных и (C) интегрированный сервис со сложной моделью. Для каждого укажите потребность в данных, ожидаемые инженерные риски, возможность тестирования и минимальную инфраструктуру. Таблица ниже задаёт форму отчёта, а не готовый рейтинг.

Вариант	Данные и зависимости	Проверка	Основной риск
Правила	Фиксированные условия	Набор тест-кейсов	Ограниченная гибкость
ML-модуль	Версионированные признаки	Offline и API tests	Drift и skew
Сложный сервис	Несколько компонентов	End-to-end и нагрузка	Каскадный отказ

Проверка решения. Подготовьте две позитивные и две негативные пробы; задайте метрику успеха, базовый сценарий и критерий остановки внедрения. Обсудите, какие допущения останутся непроверенными даже после удачного теста.

6. Самостоятельная работа: индивидуальные проектные задания

Для каждого задания студент создаёт отдельный небольшой артефакт: схему, набор тестов, код либо таблицу сравнения, прикладывает наблюдаемый результат и объясняет ограничения.

1. **DL как компонент системы.** Спроектируйте вход, тензорные преобразования, сеть, постобработку и API для изображения. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

2. **Архитектуры по модальности.** Сравните трудоемкость MLP и CNN на изображении 224x224. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

3. **Обучение на GPU.** Проверьте, что модель и batch находятся на одном устройстве, включая labels. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

4. **Функция потерь и данные.** Для 3 классов определите формы tensor logits (N,3) и target (N,). Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

5. **Цикл обучения.** Объясните роль model.eval() для Dropout и BatchNorm. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

6. **Числовая устойчивость.** Добавьте проверку finiteness loss и градиентов до сохранения checkpoint. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

7. **Регуляризация.** Исследуйте gap между training и validation loss для пяти эпох. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

8. **Transfer learning.** Замените финальный классификатор ResNet18 и обсудите fine-tuning с различными learning rates. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

9. **Checkpointing.** Сравните модель последней эпохи и лучшую по val F1. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

10. **Репродуцируемость и версии.** Составьте manifest с data_hash, commit, torch_version и best_epoch. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

11. **Инференс и вычислительные ограничения.** Сравните batch size 1 и 32 по latency на объект и общему throughput. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

12. **Документирование DL-модели.** Заполните раздел ограничения по плохому освещению для классификатора изображений. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

7. Вопросы для текущего и итогового контроля

1. В чём назначение и ограничения принципа «DL как компонент системы»?

2. Как проверить на практике утверждение: «Спроектируйте вход, тензорные преобразования, сеть, постобработку и API для изображения.»?

3. В чём назначение и ограничения принципа «Архитектуры по модальности»?

4. Как проверить на практике утверждение: «Сравните трудоемкость MLP и CNN на изображении 224x224.»?

5. В чём назначение и ограничения принципа «Обучение на GPU»?
6. Как проверить на практике утверждение: «Проверьте, что модель и batch находятся на одном устройстве, включая labels.»?
7. В чём назначение и ограничения принципа «Функция потерь и данные»?
8. Как проверить на практике утверждение: «Для 3 классов определите формы tensor logits (N,3) и target (N,).»?
9. В чём назначение и ограничения принципа «Цикл обучения»?
10. Как проверить на практике утверждение: «Объясните роль model.eval() для Dropout и BatchNorm.»?
11. В чём назначение и ограничения принципа «Числовая устойчивость»?
12. Как проверить на практике утверждение: «Добавьте проверку finiteness loss и градиентов до сохранения checkpoint.»?
13. В чём назначение и ограничения принципа «Регуляризация»?
14. Как проверить на практике утверждение: «Исследуйте gap между training и validation loss для пяти эпох.»?
15. В чём назначение и ограничения принципа «Transfer learning»?
16. Как проверить на практике утверждение: «Замените финальный классификатор ResNet18 и обсудите fine-tuning с различными learning rates.»?
17. В чём назначение и ограничения принципа «Checkpointing»?
18. Как проверить на практике утверждение: «Сравните модель последней эпохи и лучшую по val F1.»?
19. В чём назначение и ограничения принципа «Репродуцируемость и версии»?
20. Как проверить на практике утверждение: «Составьте manifest с data_hash, commit, torch_version и best_epoch.»?
21. В чём назначение и ограничения принципа «Инференс и вычислительные ограничения»?

22. Как проверить на практике утверждение: «Сравните batch size 1 и 32 по latency на объект и общему throughput.»?
23. В чём назначение и ограничения принципа «Документирование DL-модели»?
24. Как проверить на практике утверждение: «Заполните раздел ограничения по плохому освещению для классификатора изображений.»?
25. Почему высокое offline-качество не гарантирует надежность промышленной AI-системы?
26. Какие артефакты нужны для воспроизводимости и анализа инцидента?

8. Глоссарий ключевых понятий

DL как компонент системы
Нейронная сеть выполняет функцию в более широком pipeline: обучение, checkpoint, загрузка, inference и мониторинг.

Архитектуры по модальности
MLP подходит для базовых dense-признаков, CNN для локальных паттернов изображений, Transformer для последовательностей и токенов.

Обучение на GPU
Перенос данных на accelerator полезен при достаточной вычислительной нагрузке; небольшие задачи могут работать быстрее на CPU.

Функция потерь и данные
Размерности logits и целевых меток должны соответствовать постановке; CrossEntropyLoss ожидает logits, а не уже примененный softmax.

Цикл обучения
Train mode, zero_grad, forward, loss, backward, optimizer step образуют цикл обновления.

Числовая устойчивость
Взрывающиеся градиенты, переполнение softmax и неподходящий learning rate приводят к NaN.

Регуляризация
Dropout, weight decay, data augmentation и early stopping по-разному изменяют обобщение модели.

Transfer learning
Предобученная модель позволяет переиспользовать представления, но требует адаптации входных преобразований и выходного слоя.

Checkpointing
Сохраняют веса, метрики, архитектуру, конфигурацию и иногда состояние optimizer.

Репродуцируемость и версии
Указывают seed, версии torch, device, доступные операции и набор данных.

Инференс и вычислительные ограничения

Batching и precision могут повысить throughput, но влияют на память, задержки и численную точность.

Документирование DL-модели

Model card сообщает назначение, данные, риск, ограничения и протокол теста.

9. Итоги и связь со следующей темой

Представленные методы имеют смысл как части целостного управляемого жизненного цикла. Ключевой результат инженерного проектирования состоит не в максимальном числе компонентов или сложности модели, а в убедительной связи требований, проектных решений, измерений и механизмов сопровождения. Дальнейшее изучение курса развивает эти связи на следующем этапе жизненного цикла.

Список литературы

[1] Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. https://www.deeplearningbook.org/.

[2] Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 3rd ed., O'Reilly Media, 2022. https://www.oreilly.com/library/view/hands-on-machine-learning/9781098125967/.

[3] PyTorch Foundation. PyTorch documentation. https://pytorch.org/docs/stable/.

[4] Huyen C. Designing Machine Learning Systems. O'Reilly Media, 2022. https://www.oreilly.com/library/view/designing-machine-learning/9781098107956/.

[5] Sculley D. et al. Hidden Technical Debt in Machine Learning Systems. NeurIPS, 2015. https://research.google/pubs/hidden-technical-debt-in-machine-learning-systems/.

[6] Mitchell M. et al. Model Cards for Model Reporting. FAT*, 2019. https://arxiv.org/abs/1810.03993.