КАЗАХСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ
ИМЕНИ АЛЬ-ФАРАБИ

ЛЕКЦИЯ № 10

Базы данных и наборы данных при обучении искусственного интеллекта. Распознавание заболеваний растений агророботами

Бельгибаев Бауржан Абдрахимович
Д.т.н., доцент · Кафедра «Искусственный интеллект и big data»
«Технологии в интеллектуальных системах управления в IoT» · 4 курс, бакалавриат

1 Цель лекции

Сформировать системное представление о жизненном цикле данных для интеллектуальных систем управления, научить обосновывать выбор источников и структуру наборов изображений растений, организовывать обучение, тестирование и внедрение модели распознавания болезней в роботизированную IoT-платформу.

2 План лекции

1. Роль данных в интеллектуальных системах управления и агроробототехнике
2. Базы данных, датасеты и источники изображений заболеваний растений
3. Сбор, разметка, очистка и организация данных
4. Разделение выборок, дисбаланс и предотвращение утечек
5. Подготовка изображений и аугментация
6. Архитектуры CNN и перенос обучения
7. Метрики оценки и экспериментальная верификация
8. Развёртывание модели на Raspberry Pi и интеграция с IoT
9. Практический пример и исследовательский протокол

1 Роль данных в интеллектуальных системах управления

В интеллектуальной системе управления данные образуют основу для восприятия состояния объекта, формирования решения и последующего воздействия. В традиционном контуре обратной связи контроллер получает числовые сигналы датчиков; в агророботе к ним добавляются изображения листьев, координаты растения, освещённость, влажность и температура. Алгоритм машинного обучения преобразует наблюдение в оценку состояния, например вероятность наличия определённого заболевания.

Типовая цепочка обработки: камера и датчики → сбор и синхронизация → предварительная обработка → модель компьютерного зрения → оценка уверенности → логика принятия решения → исполнительное устройство / уведомление оператору → журнал результатов. Предсказание модели не должно автоматически приравниваться к подтверждённому фитопатологическому диагнозу.

Пусть x_t — вектор признаков в момент t, θ — параметры модели, $\hat{y}_t = f\theta(x_t)$ — её предсказание, а $u_t = \pi(\hat{y}_t, z_t)$ — управляющее воздействие с учётом дополнительных ограничений z_t. Такая архитектура позволяет отделить вероятностную оценку болезни от безопасной логики управления роботом.

2 Базы данных и наборы данных

База данных (database) — организованное хранилище записей, метаданных и связей; набор данных (dataset) — подготовленная совокупность наблюдений, предназначенная для анализа, обучения или проверки модели. Изображения могут храниться в файловой системе или объектном хранилище, а метаданные — в SQL/NoSQL базе. Для воспроизводимости исследования необходимы идентификаторы, версия набора и описание происхождения изображений.

Для обучения распознаванию болезней растений часто рассматривают PlantVillage как пример набора размеченных изображений листьев. Однако фотографии на относительно однородном фоне не полностью отражают условия реального поля: сложный фон, тени, частичные перекрытия, разные камеры и стадии развития заболевания. Поэтому полевое тестирование на независимых данных обязательно.

2 Базы данных и наборы данных (продолжение)

Поле метаданных	Назначение
image_id	Уникальный идентификатор изображения
plant_species	Вид или культура растения
disease_label	Класс болезни / здоровое растение
plant_id, plot_id	Идентификаторы растения и участка для корректного разделения
timestamp, location	Время и место съёмки (при допустимости хранения)
camera, illumination	Камера и условия освещения
annotator, confidence	Эксперт-разметчик и уверенность
dataset_version	Версия набора данных

2 Базы данных и наборы данных (продолжение)

Разметка для классификации задаёт одну или несколько меток всему изображению; для детекции — ограничивающие прямоугольники вокруг поражённых областей; для сегментации — маску пикселей. Выбор типа разметки определяется конечной задачей: обнаружить факт заболевания, локализовать повреждение или оценить площадь поражения.

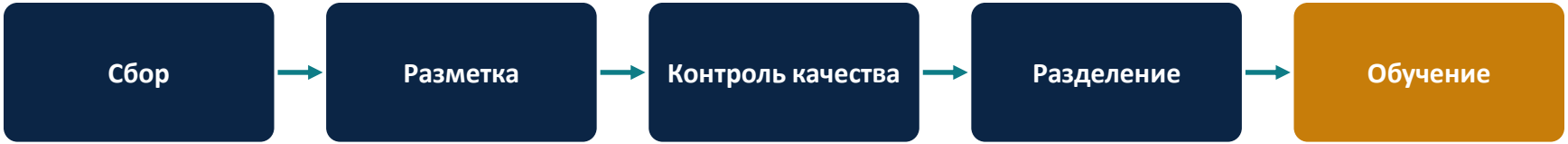
3 Сбор, разметка и контроль качества данных

Сбор изображений следует планировать с учётом культуры, сорта, стадии роста, времени суток, угла съёмки и условий освещения. Полезно включать отрицательные примеры: здоровые листья, механические повреждения, дефицит питания и признаки, похожие на болезнь. Без таких примеров модель может переобучиться на случайные признаки фона или освещения.

Контроль качества включает проверку формата и разрешения, выявление повреждённых файлов, удаление точных дубликатов, поиск почти одинаковых изображений, проверку согласованности меток и экспертный аудит спорных случаев. Если одно растение снято серией кадров, эти кадры не должны попадать одновременно в обучение и тестирование: иначе качество будет завышено из-за утечки данных.

Для экспертной согласованности можно использовать коэффициент Каппа Коэна: $\kappa = (p_o - p_e)/(1 - p_e)$, где p_o — наблюдаемая доля совпадений разметчиков, p_e — ожидаемая доля случайных совпадений. Значение κ следует интерпретировать в контексте распределения классов и сложности диагностики.

5 Жизненный цикл набора данных



Типичное разделение выборок (пример)



Рис. 18. Тестовая выборка не должна использоваться при подборе параметров модели.

4 Разделение выборок и дисбаланс классов

Набор данных разделяют на train (обучение), validation (подбор гиперпараметров) и test (окончательная независимая оценка). Например, 70/15/15 % — допустимая отправная пропорция, но она не является универсальной. Критически важно разделять данные по растению, участку, сессии съёмки или хозяйству, а не случайно по кадрам, если кадры коррелированы.

Для проверки обобщающей способности желательно выделить внешний тестовый набор из другого хозяйства, сезона или камеры. Стратификация по классам помогает сохранять их представленность, но не заменяет группового разделения. Все преобразования, использующие статистику данных, включая нормализацию по среднему и дисперсии, настраиваются только на обучающей части.

При дисбалансе классов можно применять взвешенную функцию потерь, контролируемую аугментацию меньшинства, балансировку батчей и сбор дополнительных полевых примеров. Избыточное копирование изображений не создаёт новой информации и не гарантирует улучшения на независимом тесте.

5 Предобработка изображений и аугментация

Типовая подготовка включает декодирование, проверку цветового пространства, изменение размера, нормализацию значений пикселей и формирование тензора. Если модель обучалась на RGB-изображениях размером 224 × 224, при внедрении необходимо воспроизвести тот же порядок каналов и параметры нормализации. Библиотека OpenCV по умолчанию читает цветные изображения в BGR, что требует явного преобразования при использовании RGB-моделей.

Нормализация канала: $x' = (x - \mu)/\sigma$, где μ и σ вычислены по обучающей выборке либо взяты из спецификации предобученной модели. Аугментации могут включать небольшие повороты, изменения яркости, масштабирование и отражения, если они не искажают диагностически важные признаки. Аугментацию применяют только к обучающей выборке.

При слишком сильной коррекции цвета модель может перестать различать симптомы, связанные с пожелтением или потемнением тканей. Поэтому диапазоны аугментации следует обосновывать агрономическими и оптическими условиями реального применения.

6 Архитектура CNN и перенос обучения

Свёрточная нейронная сеть извлекает локальные визуальные признаки с помощью фильтров. Для входной карты X и ядра K свёртка записывается как $Y(i,j) = \Sigma_m \Sigma_n K(m,n)X(i+m,j+n) + b$. Нелинейность ReLU задаётся как $\max(0,z)$. Последние слои формируют оценки классов; softmax переводит логиты z_k в распределение $p_k = \exp(z_k)/\Sigma_j \exp(z_j)$.

Для многоклассовой классификации применяется перекрёстная энтропия: $L = -(1/N)\ \Sigma_i \Sigma_k\ y_{ik} \log(p_{ik})$, где y_{ik} — истинная метка, p_{ik} — предсказанная вероятность. При дисбалансе классов можно использовать веса w_k внутри суммы. Оптимизация выполняется, например, алгоритмом Adam; выбор скорости обучения и регуляризации производится по validation.

Перенос обучения (transfer learning) позволяет использовать предварительно обученную MobileNet, EfficientNet или другую компактную CNN. На первом этапе замораживают часть извлекателя признаков и обучают классификатор; затем при необходимости размораживают верхние слои с меньшей скоростью обучения. Для Edge AI особенно важны размер модели, задержка, потребление памяти и возможность квантования.

Свёрточная сеть и перенос обучения

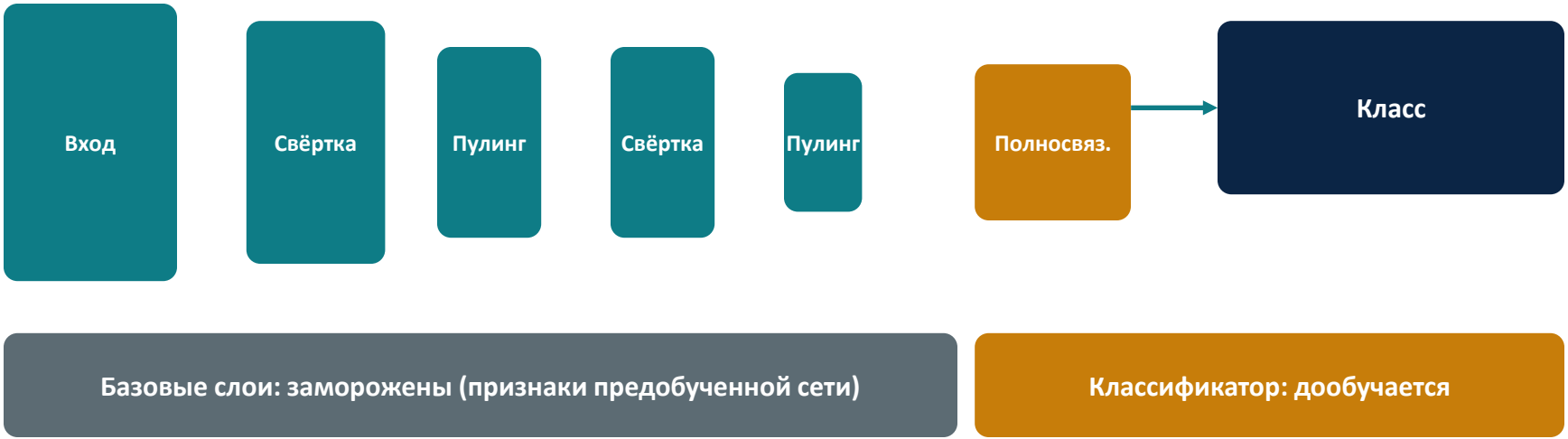


Рис. 19. Перенос обучения сокращает объём данных и время обучения.

7 Метрики качества и верификация

Для бинарного класса «болезнь» определяются TP — правильно обнаруженная болезнь, FP — ложное срабатывание, FN — пропущенная болезнь, TN — правильно определённое отсутствие болезни. Accuracy = (TP+TN)/(TP+TN+FP+FN); Precision = TP/(TP+FP); Recall = TP/(TP+FN); Specificity = TN/(TN+FP); F1 = 2·Precision·Recall/(Precision+Recall).

Accuracy может вводить в заблуждение при редких заболеваниях: модель, всегда предсказывающая здоровое растение, покажет высокую общую точность при крайне низкой чувствительности. Поэтому необходимо представлять матрицу ошибок, macro-F1, чувствительность по каждому классу и доверительные интервалы. Для систем с дорогостоящими последствиями пропуска болезни Recall может быть важнее общей Accuracy.

Пример: TP = 84, FP = 12, FN = 16, TN = 188. Тогда Precision = 84/96 = 0,875; Recall = 84/100 = 0,84; F1 ≈ 0,857; Accuracy = 272/300 ≈ 0,907. Это иллюстративные расчётные данные, а не результаты эксперимента.

7 Метрики качества и верификация (продолжение)

Показатель	Что измеряет	Ограничение
Macro-F1	Среднее F1 по классам	Чувствителен к редким классам
Recall	Доля найденных положительных случаев	Не отражает ложные тревоги
Precision	Доля верных положительных срабатываний	Не отражает пропуски
Latency p95	95-й процентиль задержки	Требует измерения на целевом устройстве
FPS	Число обработанных кадров в секунду	Не характеризует точность распознавания

8 Развёртывание на Raspberry Pi и связь с IoT

На Raspberry Pi модель может работать локально с использованием оптимизированного runtime, а результат публиковаться в MQTT или передаваться в серверную систему. Edge-инференс уменьшает зависимость от облачного соединения и позволяет быстро формировать тревогу. Однако вычислительные ограничения требуют измерять задержку полного контура: захват кадра, предобработка, инференс, постобработка и доставка сообщения.

Полная задержка: T_{e2e} = Tcapture + Tpre + Tinfer + Tpost + Tnetwork + Tdecision. Для безопасного управления важно не только среднее значение, но и верхние процентили задержки, устойчивость к пропускам кадров и поведение при отказе сети.

В системе управления результат модели следует использовать как вход в проверяемую логику. Например, робот может помечать координату подозрительного растения и передавать снимок оператору для подтверждения. Автоматическое опрыскивание требует дополнительных правил безопасности, учёта погоды, разрешений и достоверности диагноза.

9 Практический пример: классификация листьев растений

Исследовательская постановка: создать прототип распознавания здоровых и поражённых листьев на компактной платформе, сравнить качество модели на внутреннем и внешнем тестах и оценить задержку на Raspberry Pi. Для учебного примера ниже предполагается структура каталогов train/healthy, train/diseased, val/healthy, val/diseased и test/healthy, test/diseased. Разделение выполняется заранее по идентификатору растения, а не случайно по отдельным фотографиям.

Пример Python: обучение классификатора на основе MobileNetV3

```
import torch

from torchvision import datasets, models, transforms

from torch import nn

from torch.utils.data import DataLoader

train_tf = transforms.Compose([
```

Практический пример: классификация листьев растений (продолжение)

```
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])
```

val_tf = transforms.Compose([

```
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])
```

Практический пример: классификация листьев растений (продолжение)

```
train_ds = datasets.ImageFolder('data/train', transform=train_tf)
val_ds = datasets.ImageFolder('data/val', transform=val_tf)
train_loader = DataLoader(train_ds, batch_size=16, shuffle=True)
val_loader = DataLoader(val_ds, batch_size=16, shuffle=False)
model = models.mobilenet_v3_small(weights=models.MobileNet_V3_Small_Weights.DEFAULT)
for param in model.features.parameters():
    param.requires_grad = False
model.classifier[3] = nn.Linear(model.classifier[3].in_features, len(train_ds.classes))
```

Практический пример: классификация листьев растений (продолжение)

optimizer = torch.optim.Adam(model.classifier.parameters(), lr=1e-3)

criterion = nn.CrossEntropyLoss()

```
for epoch in range(5):
    model.train()
    for images, labels in train_loader:
        optimizer.zero_grad()
        loss = criterion(model(images), labels)
        loss.backward()
        optimizer.step()
    model.eval()
```

Практический пример: классификация листьев растений (продолжение)

```
correct = total = 0
with torch.no_grad():
    for images, labels in val_loader:
        predictions = model(images).argmax(dim=1)
        correct += (predictions == labels).sum().item()
        total += labels.size(0)
print(epoch + 1, 'val accuracy:', correct / max(total, 1))
```

Практический пример: классификация листьев растений (продолжение)

Код демонстрирует базовый процесс обучения и оценки, но не заменяет полный исследовательский протокол. Для научной публикации дополнительно необходимы фиксация random seed, сохранение лучшей модели по validation, независимое тестирование, матрица ошибок, анализ ошибок разметки и воспроизводимые сведения о версиях программных библиотек.

План эксперимента для докторантов

Этап 1. Сформировать реестр данных: классы, источники, лицензии, культура, условия съёмки и версии.

Этап 2. Разделить изображения по растениям/участкам, зафиксировать тестовую выборку до обучения.

Этап 3. Сравнить базовую CNN и модель с переносом обучения при одинаковых разбиениях.

Этап 4. Оценить macro-F1, Recall по каждому классу и матрицу ошибок.

Этап 5. Протестировать на независимых полевых снимках и описать сдвиг домена.

Этап 6. Измерить задержку и память на Raspberry Pi, повторить измерения не менее 30 раз.

Этап 7. Подготовить протокол внедрения: порог уверенности, отказоустойчивость, журнал событий и подтверждение оператором.

10 Научно-исследовательские проблемы и направления развития

Основная научная проблема — переносимость моделей из контролируемой лабораторной среды в реальные агротехнические условия. Доменные различия возникают из-за освещения, фона, камеры, сорта растения и стадии заболевания. Перспективные направления включают адаптацию к домену, самоконтролируемое обучение, активный отбор снимков для экспертной разметки, оценку неопределённости и мультимодальное объединение изображения с микроклиматическими измерениями.

Для управления важно различать диагностическую точность и эффективность действия: высокая F1 сама по себе не гарантирует снижение потерь урожая. Требуется проверять, как решения системы влияют на время обследования, частоту ложных тревог, нагрузку оператора и агрономический результат в контролируемом исследовании.

13 Заключение

Наборы данных определяют границы применимости интеллектуальной модели. Для достоверного распознавания заболеваний растений необходимо обеспечить репрезентативный сбор и экспертную разметку, исключить утечки между выборками, оценивать модель по нескольким метрикам и проверять её на независимых полевых данных. При интеграции в IoT-агроробота к точности добавляются требования к задержке, энергопотреблению, устойчивости связи и безопасному принятию решений.

14 Контрольные вопросы

1. Чем база данных отличается от набора данных машинного обучения?
2. Какие метаданные следует хранить для изображений листьев?
3. Чем классификация отличается от детекции и сегментации?
4. Почему изображения одного растения нельзя делить между train и test?
5. Какие риски возникают при использовании лабораторных снимков в полевых условиях?
6. Как выявляют ошибки и несогласованность экспертной разметки?
7. Как выполняется разделение train/validation/test?
8. Какие методы позволяют уменьшить влияние дисбаланса классов?
9. Как нормализация и аугментация влияют на обучение?
10. Как работает свёрточный слой CNN?
11. В чём преимущества переноса обучения для Edge AI?

14 Контрольные вопросы (продолжение)

12. Почему Accuracy недостаточна при редких заболеваниях?
13. Как рассчитываются Precision, Recall и F1?
14. Из каких составляющих складывается задержка IoT-системы распознавания?
15. Какие требования предъявляются к верификации модели перед использованием в агророботе?

15 Рекомендуемая литература и ресурсы

1. **Ли П. Архитектура интернета вещей. М.:** ДМК Пресс, 2018.

2. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016.

3. **Szeliski R. Computer Vision:** Algorithms and Applications. 2nd ed. Springer, 2022.

4. **PyTorch Documentation:** https://pytorch.org/docs/stable/

5. **Torchvision Documentation:** https://pytorch.org/vision/stable/

6. **OpenCV Documentation:** https://docs.opencv.org/

7. **Raspberry Pi Documentation:** https://www.raspberrypi.com/documentation/

8. PlantVillage project and associated scientific publications — проверять условия лицензирования и состав конкретной версии набора.