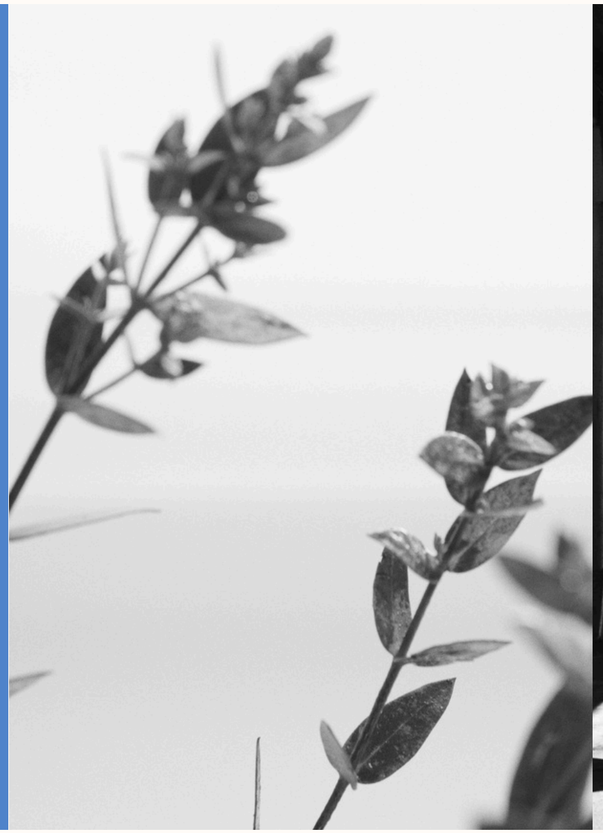
Al-Farabi Kazakh National University

# УНИВЕРСИТЕТТЕГІ ҒИМАРАТТАР МЕН ҚҰРЫЛЫСТАРДЫ ТАНУ

Компьютерлік ғылымдар

# МАҚСАТЫ

- Университеттегі әртүрлі ғимараттарды суреттер арқылы автоматты түрде танитын жүйе құру
- Деректерді өңдеу, модель таңдау және оқыту арқылы компьютерлік көру (computer vision)
- Алдын ала дайындалған нейрондық желілерді (EfficientNet, MobileNet, ResNet) қолдана отырып, нақты нәтижелерге қол жеткізу
- Ғимараттарды тану процесін визуализациялау арқылы пайдаланушыға ыңғайлы интерфейс жасау



# ДЕРЕКТЕР ТУРАЛЫ

**KazNU**

| | |
|---|---|
| 01 | library |
| 02 | rektorat |
| 03 | zholdasbekov |
| 04 | keremet |
| 05 | internet |

Сурет саны: Әр категорияда шамамен 100 сурет
Барлығы ~500 ден аса сурет

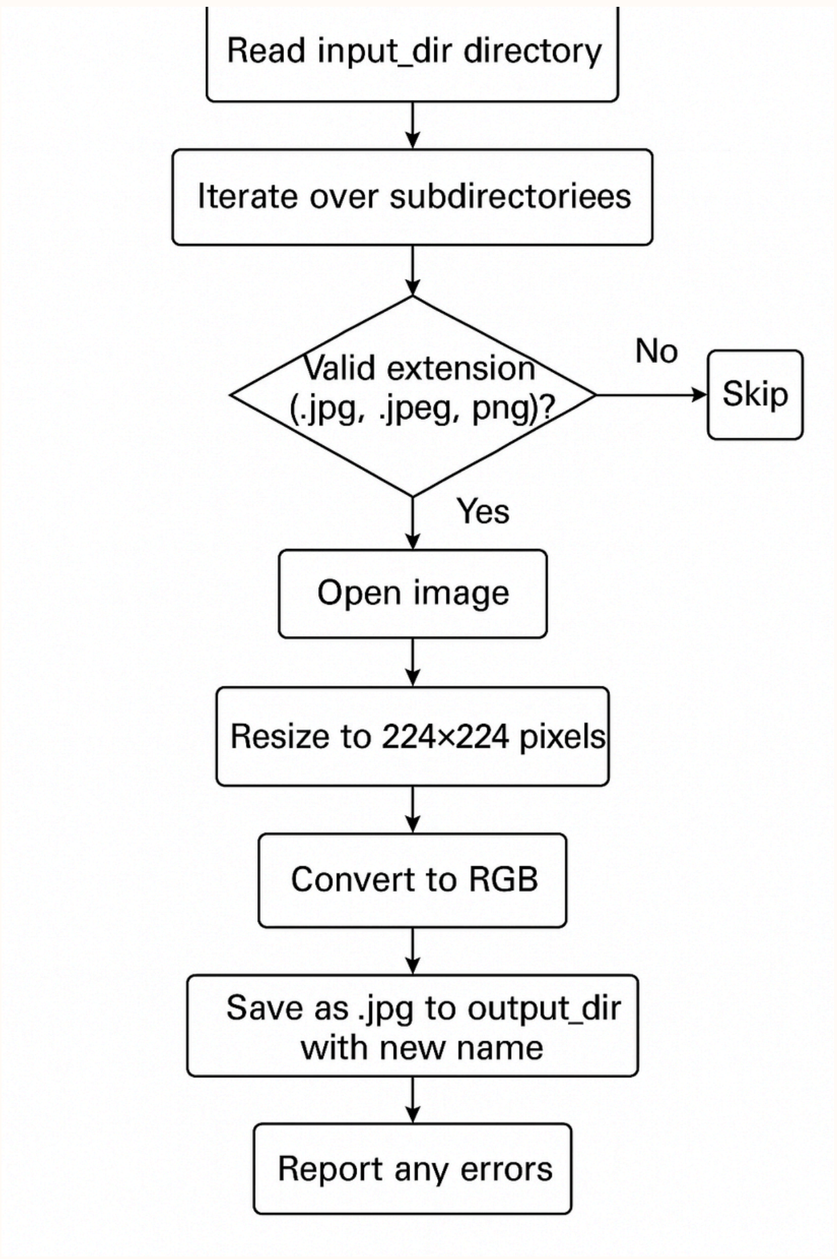
Форматтар: .jpg, .jpeg, .png (аралас түрде)

Өлшемдер: әртүрлі барлығын 224*224 дегендей қылып өзгерту керек

Сапа: кейбір суреттер бұлыңғыр, жарығы әлсіз

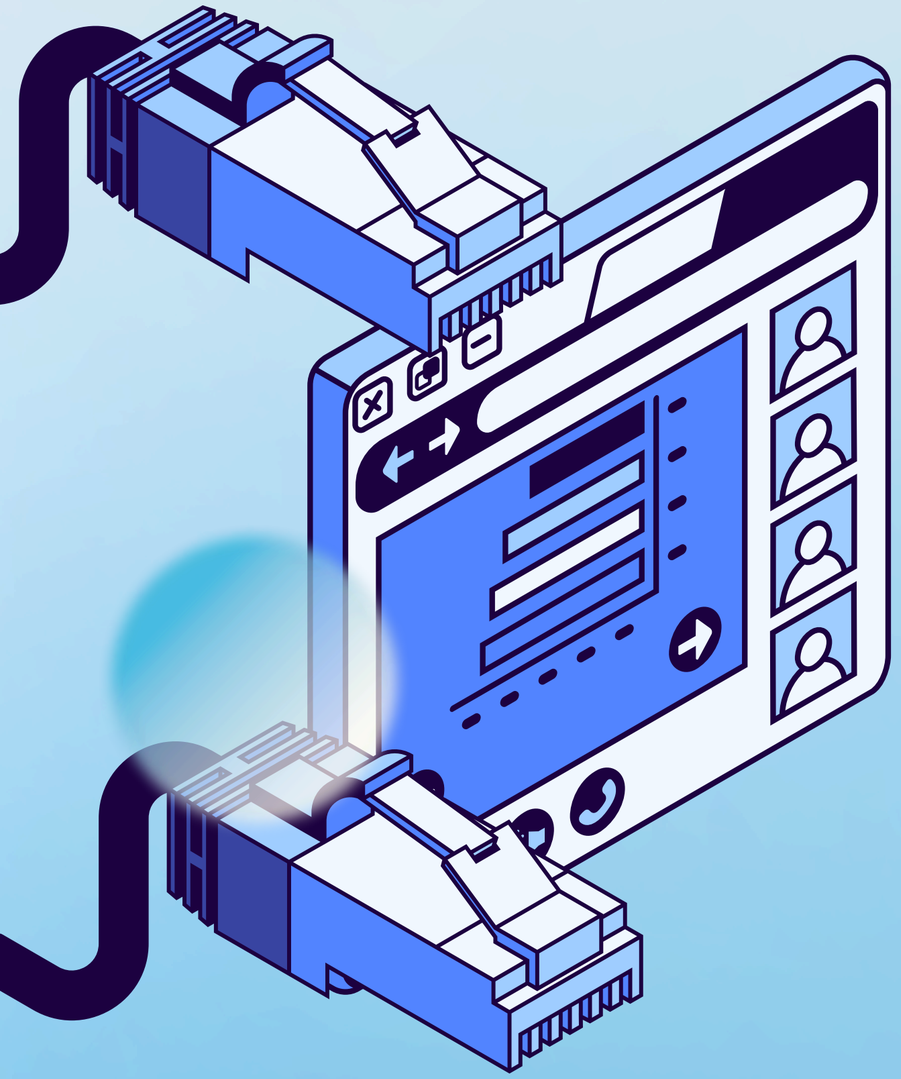
Мәселелер:
Формат пен өлшем біркелкі емес
Модельге беру үшін алдын ала өңдеуді қажет етеді

Суреттерді қайта өлшемдеп және оларды жаңа форматта сақтайтын бағдарлама. Алдымен, көрсетілген input_dir қалтасында орналасқан барлық ішкі қалталарды сканерлейді. Әрбір ішкі қалта суреттерді сақтайды, және код әрбір қалтаның ішіндегі файлдарды оқиды. Файлдардың кеңейтімдері тексерілып, тек .jpg, .jpeg, немесе .png форматындағы суреттер өңделеді. Басқа форматтағы суреттер өткізіліп қалады. Содан кейін, әрбір жарамды сурет ашылып, оның өлшемі 224x224 пиксельге өзгертіледі, бұл нейрондық желілерге сәйкес келетін өлшем. Суреттер RGB форматында конвертацияланып, жаңа атпен және .jpg форматында output_dir қалтасына сақталады. Әрбір суреттің жаңа атауы бұрынғы атпен сәйкес келеді, тек .jpg кеңейтімі қосылады. Егер қандай да бір суретті ашу немесе өңдеу кезінде қате туындаса, бағдарлама бұл қатені хабарлап, қалған суреттерді өңдеуді жалғастырады.



KazNU университетінің суреттер деректер жиынтығын пайдаланып, суреттерді классификациялау үшін CNN (Convolutional Neural Network) моделін құрады және оны оқыту мен тестілеу процесінде қолданады. Алдымен, KazNU_Dataset классы арқылы суреттер мен олардың белгілері жүктеледі және әр суретке қажетті түрлендірулер (Resize, ToTensor, Normalize) қолданылады. Содан соң, CNN моделі үш конволюциялық қабаттан, макс-пуллинг қабаттарынан және екі толық байланыстырушы қабаттан тұрады. Модельді оқыту барысында, шығынды минимизациялау үшін CrossEntropyLoss функциясы қолданылады, ал оптимизация үшін Adam әдісі пайдаланылады. 10 кезең бойы модель оқытылып, әр кезеңде дәлдік пен шығын есептелiп, нәтижелер графиктер арқылы визуализацияланады. Оқытудан кейін тестілеу деректер жиынтығында модельдің соңғы дәлдігі есептеледі.

# 🛠 Қолданылған әдістер



**1. Трансферлік оқыту (Transfer Learning)**
Модель ретінде алдын ала **ImageNet** датасетінде оқытылған **ResNet50** архитектурасы таңдалды. Бұл әдіс модельдің бастапқы қабаттарын қайта оқытпай-ақ, тек соңғы қабатын бейімдеу арқылы жаңа тапсырмаларға қолдануға мүмкіндік береді. Осы жоба аясында:

- Соңғы **fully connected** қабат **resnet.fc** алынбақшы болып, жаңа сыныптар санына бейімделді.
- Бұл тәсіл аз көлемдегі деректерде де жоғары дәлдікке қол жеткізуге мүмкіндік береді.

**2. Кастом датасет (Custom Dataset)**
**PyTorch** кітапханасының **Dataset** класын мұрагерлікке алып, арнайы **KazNU_Dataset** класы құрылды. Бұл кластың негізгі міндеттері:

- Суреттер мен олардың меткаларын автоматты түрде оқу;
- Кастом трансформация қолдану;
- Модельге оқу үшін қажетті `__getitem__` және `__len__` әдістерін анықтау.

### 3. Деректерді трансформациялау

Суреттер модельге берілмес бұрын **ResNet50** моделімен үйлесімді болу үшін келесі өңдеулерден өтеді:

- Өлшемді өзгерту
- Қалыпқа келтіру (**normalization**)
- **Tensor** форматына келтіру

Бұл трансформациялар **ResNet50_Weights.DEFAULT.transforms()** арқылы автоматты түрде алынды.

### 4. Оқыту процесі (Training Loop)

Модельдің параметрлері келесі тәсілдермен оқытылды:

- **Loss** функциясы: **CrossEntropyLoss** — классификация үшін стандартты функция.
- Оптимизатор: **Adam** — градиенттерді тиімді және тез есептейтін алгоритм.
- Әр эпох сайын модель:
  - Тренинг деректеріндегі шығын (**loss**) және дәлдік (**accuracy**) есептейді;
  - Қайталанатын графиктерде нәтижелерді визуализациялайды.

# ОПТИМИЗАТОРЛАР

Adam оптимизаторы кодта мына себептермен қолданылды. Біріншіден, ол әрбір параметр үшін жеке оқыту қарқынын (learning rate) бейімдеп есептейді, яғни әр қабаттағы салмақтарға сәйкес жылдамдықты автоматты түрде реттейді. Екіншіден, Adam градиенттің бірінші және екінші моменттерін (орташа және дисперсия) ескере отырып жұмыс істейді, сондықтан тербелістер азайып, конвергенция үдейді. Үшіншіден, жаңа жобада немесе деректер жиыны әртүрлі болғанда, Adam «қораптан шығарған күйінде» (default параметрлерімен) әдетте жақсы нәтиже береді. Төртіншіден, гиперпараметрлерді (learning rate, $\beta_1$, $\beta_2$) аса өзгертпей-ақ тиімді жаттықтыруға болады.

Егер модель өте терең және үлкен деректерде жаттықтырылатын болса, қарапайым SGD+momentum қолданып көруге болады, бірақ онда оқыту қарқынын мұқият таңдау керек. RMSprop адаптивті әдістердің бірі ретінде градиентті квадраттарының орташа мәнін ескереді, сондықтан кейде Adam орнына жеңілдеу эксперимент үшін лайық келеді. AdamW – Adam-ның weight decay (L2-регуляризация) қолдауы жақсартылған нұсқасы, ол overfitting-ті азайтуға көмектеседі. Егер модель біркелкі әрі тұрақты оқу керек болса және деректер шуылы аз болса, SGD+momentum жақсы генерализация береді. Кішкентай деректер жинағында RMSprop қарапайым және жылдам нәтиже көрсетуі мүмкін. Ал егер сен тек гиперпараметрлерді аз оқып, жылдам прототип жасағың келсе, Adam ең оңай және әмбебап таңдауды ұсынады

# ИНТЕРФЕЙС

Gradio – Python функцияларын тез арада веб-интерфейс түрінде көрсетуге арналған құрал. Әуелі сіз функцияңыздың қандай типтегі кіріс пен шығыс алатынын анықтайсыз (мысалы, сурет, мәтін, график және т. с. с.). Gradio осы сипаттамаларға сүйене отырып, автоматты түрде HTML+JavaScript негізіндегі фронтенд генерациялайды. Қолданушы браузерде пайда болған бетте қажетті элементтерді (батырмалар, жүктеу өрістері, слайдерлер) пайдаланып деректерді енгізеді. Енгізілген деректер серверге жіберіліп, сіз көрсеткен Python функциясына аргумент ретінде түседі. Функция есептеп шыққан нәтиже (мысалы, аннотацияланған сурет немесе ықтималдықтар) Gradio арқылы қайта фронтендке жеткізіліп, сәйкес виджеттерде көрсетіледі. Барлық бұл процесс – кіріс → функция → шығыс – Gradio-ның event loop-ы мен API-інің арқасында бір жолмен жүзеге асады. Осылайша, код жазбай-ақ, бірнеше минут ішінде интерактивті демо немесе прототип жасауға болады.