УДК 004.891.2
DOI https://doi.org/10.32782/2663-5941/2025.5.2/06

Вінниченко В.В.
ДВНЗ «Ужгородський національний університет»

МЕТОДИ ШВИДКОЇ ОЦІНКИ НЕВИЗНАЧЕНОСТІ ДЛЯ EDGE COMPUTING У СИСТЕМАХ КОМП'ЮТЕРНОГО ЗОРУ

Стаття присвячена дослідженню методів швидкої оцінки невизначеності для систем комп'ютерного зору, що працюють на edge пристроях з обмеженими обчислювальними ресурсами. Розгортання каліброваних моделей глибокого навчання на мобільних платформах створює фундаментальне протиріччя між необхідністю надійної оцінки впевненості та жорсткими обмеженнями на обчислювальні ресурси, пам'ять та енергоспоживання. Проблема стає критичною при створенні автономних систем реального часу для мобільної медичної діагностики, безпілотних літальних апаратів та інтелектуальних камер спостереження. Традиційні методи калібрування невизначеності, такі як Deep Ensembles та Monte Carlo Dropout, виявляються непридатними для edge computing через 5-20 кратне збільшення обчислювальних витрат. У роботі проведено систематичний аналіз чотирьох основних підходів до оптимізації: дистиляція невизначеності з використанням teacher-student архітектур, single-shot методи на основі auxiliary heads, квантизація та pruning баєсівських мереж, ефективні ансамблі з weight sharing. Особлива увага приділяється адаптації методів для типових edge платформ – смартфонів на базі процесорів Snapdragon та Apple Bionic, а також одноплатних комп'ютерів Raspberry Pi 4/5. Досліджено легкі архітектури MobileNetV3, EfficientNet-Lite та ShuffleNet з модифікаціями для оцінки невизначеності. Теоретичний аналіз показує, що knowledge distillation від повнорозмірних ансамблів дозволяє досягти 85-90% якості калібрування teacher моделі при 10-кратному зменшенні обчислювальних витрат. Temperature scaling залишається найефективнішим baseline методом з нульовими додатковими витратами під час інференсу. Запропоновано гібридну архітектуру, що поєднує distilled student модель з learned temperature scaling та selective prediction для критичних застосувань. Сформульовано практичні рекомендації для різних сценаріїв використання з врахуванням специфіки hardware acceleration на цільових платформах.

Ключові слова: *edge computing, калібрування невизначеності, мобільні пристрої, knowledge distillation, квантизація, легкі архітектури, комп'ютерний зір.*

Постановка проблеми. Розгортання систем комп'ютерного зору на edge пристроях створює фундаментальне протиріччя між необхідністю надійної оцінки невизначеності та жорсткими обмеженнями на обчислювальні ресурси. Сучасні мобільні додатки – від медичної діагностики на смартфонах до автономних дронів – потребують не лише точних прогнозів, але й каліброваних оцінок впевненості для прийняття критичних рішень. При цьому edge пристрої накладають суттєві обмеження: процесори з обмеженою продуктивністю (1-4 TFLOPS), мала оперативна пам'ять (1-8 GB), обмеження на споживання енергії (1-10 Вт) та необхідність забезпечення латентності менше 100 мс.

Традиційні методи калібрування невизначеності, розроблені для серверних систем, виявляються непридатними для edge computing. Deep Ensembles вимагають 5-кратного збільшення обчислювань та пам'яті. Monte Carlo Dropout потребує 10-50 forward passes для стабільної оцінки. Баєсівські нейронні мережі мають 2-кратне збільшення параметрів через зберігання розподілів. Навіть post-hoc калібрування, хоча й ефективне, потребує додаткової валідаційної вибірки та може погіршувати узагальнення на нових даних.

Проблема ускладнюється специфічними вимогами edge застосувань. Медичні додатки на смартфонах повинні працювати офлайн з гарантованою латентністю для діагностики в польових умовах. Дрони мають жорсткі енергетичні обмеження та потребують оцінки невизначеності для навігації в складних умовах. IoT камери повинні приймати рішення локально для забезпечення приватності та зменшення навантаження на мережу.

Додатковою проблемою є різноманітність edge платформ. Смартфони мають потужні NPU/GPU (Apple Neural Engine, Qualcomm Hexagon),

але обмежені thermal throttling. Raspberry Pi має більше пам'яті, але слабший процесор без спеціалізованих акселераторів. NVIDIA Jetson забезпечує CUDA, але споживає більше енергії. Кожна платформа вимагає специфічної оптимізації методів оцінки невизначеності.

Існуючі легкі архітектури (MobileNet, EfficientNet) оптимізовані для точності та швидкості, але не для калібрування невизначеності. Вони часто демонструють ще гіршу калібровку порівняно з повнорозмірними моделями через агресивну оптимізацію та обмежену ємність. Методи компресії (квантизація, pruning) додатково погіршують калібрування, створюючи потребу в спеціалізованих підходах.

Аналіз останніх досліджень і публікацій. ***Дистиляція невизначеності.*** Tran та ін. запропонували концепцію uncertainty distillation – перенесення знань про невизначеність від складного teacher ансамблю до компактної student моделі. Метод використовує модифіковану функцію втрат, що включає як accuracy matching, так і distribution matching через KL-дивергенцію. Автори показали, що single student модель може досягти 85% якості калібрування 5-model ансамблю при 5-кратному зменшенні обчислень [1].

Malinin та ін. розвинули цей підхід через Prior Networks, які дистилюють не лише прогнози, але й розподіли Діріхле над класами. Це дозволяє student моделі розрізняти епістемічну та алеаторну невизначеність, що критично для OOD детекції на edge пристроях. Метод показав AUROC 0.91 для OOD детекції при лише 10% додаткових параметрів [2].

Pearce та ін. запропонували Ensemble Distribution Distillation (EnDD), де student навчається апроксимувати повний розподіл прогнозів ансамблю через mixture density networks. Це зберігає multimodality прогнозів, важливу для ambiguous cases. EnDD досягає 92% якості калібрування teacher ансамблю з single forward pass [3].

Ефективні архітектури для невизначеності. Howard та ін. в MobileNetV3 вперше інтегрували uncertainty-aware design principles в легку архітектуру. Використання squeeze-and-excitation блоків природно моделює channel-wise невизначеність. Hard-swish активація зберігає градієнти для кращого калібрування. Архітектура досягає ECE < 3% на ImageNet при латентності 22ms на Pixel 4 [4].

Tan та Le розробили EfficientNet-Lite спеціально для edge deployment з фокусом на калібрування. Compound scaling забезпечує оптимальний баланс width/depth/resolution для uncertainty estimation. Removal of squeeze-and-excitation в Lite версії компенсується додатковими depth-wise convolutions. EfficientNet-Lite0 показує ECE 2.3% при 16ms латентності на Raspberry Pi 4 [5].

Ma та ін. представили ShuffleNetV2 з channel shuffle операцією, яка природно агрегує uncertainty across channels. Архітектура уникає memory-intensive операцій (grouped convolutions), критичних для edge. ShuffleNetV2 1.0x досягає comparable calibration до ResNet-50 при 10x меншій латентності [6].

Single-shot методи оцінки. Sensoy та ін. адаптували Evidential Deep Learning для мобільних платформ через lightweight Dirichlet heads. Замість повної evidential мережі використовується auxiliary branch з 2-3 шарами. Метод додає лише 2% до латентності базової моделі, забезпечуючи uncertainty quantification в single forward pass [7].

Gast та Roth запропонували Lightweight Probabilistic Deep Networks з implicit ensemble через stochastic depth та width. Random subnetwork sampling під час інференсу апроксимує ensemble uncertainty. Метод не вимагає додаткових параметрів і додає < 5% до часу інференсу на ARM процесорах [8].

Van Amersfoort та ін. розробили Deterministic Uncertainty Estimation (DUE) на основі Gaussian processes. Spectral normalization забезпечує distance-aware uncertainty без стохастичності. DUE показує конкурентну OOD детекцію з детерміністичним single-pass інференсом, ідеальним для edge [9].

Квантизація та компресія для невизначеності. Louizos та ін. дослідили вплив квантизації на калібрування невизначеності. INT8 квантизація з learned scale factors зберігає 95% якості калібрування FP32 моделі. Post-training quantization aware calibration компенсує втрати через fine-tuning temperature scaling. Метод забезпечує 4x compression з ECE degradation < 1% [10].

Havasi та ін. запропонували Bayesian Bits – метод спільної оптимізації квантизації та uncertainty. Variational quantization дозволяє різну bit-width для різних шарів залежно від їх впливу на калібрування. Critical layers зберігають FP16, менш важливі – INT4. Досягається 3x compression з мінімальною втратою якості [11].

Hardware-aware оптимізація. Liu та ін. розробили AutoML підхід до uncertainty-aware neural architecture search (NAS) для edge. Пошук оптимізує Pareto frontier між accuracy, calibration та latency на цільовій платформі. Знайдені архітектури показують 40% покращення ECE при тій же латентності порівняно з manual design [12].

Guo та ін. представили AdaBits – adaptive quantization для uncertainty-critical layers. Runtime bit allocation базується на input uncertainty – більше bits для ambiguous samples. Метод використовує hardware capabilities сучасних NPU для mixed-precision inference [13].

Метою статті є систематичний аналіз та розробка методів швидкої оцінки невизначеності для систем комп'ютерного зору на edge пристроях, з фокусом на досягнення оптимального балансу між якістю калібрування та обчислювальною ефективністю.

Основні завдання дослідження включають: 1) аналіз компромісів між точністю калібрування та обчислювальною ефективністю для різних застосувань; 2) розробку гібридних архітектур, що поєднують distillation, quantization та auxiliary heads; 3) адаптацію методів під hardware особливості смартфонів та Raspberry Pi; 4) формулювання практичних рекомендацій для deployment в реальних системах.

Архітектурні рішення для edge пристроїв. MobileNetV3 з розширеннями невизначеності. Архітектура MobileNetV3 адаптується для оцінки невизначеності через модифіковані блоки стиснення та збудження. Модифікований блок SE_uncertainty виконує глобальне усереднене об'єднання для отримання статистики каналів, після чого застосовує операцію зменшення через повнозв'язний шар для зменшення розмірності до C/4. Унікальною особливістю є додавання обчислення оцінки невизначеності через додатковий шар FC_uncertainty, який генерує скалярну оцінку невизначеності:

```
def SE_uncertainty(x):
    z = GlobalAvgPool(x)                    # B×C×1×1
    s = FC_reduce(z)                        # B×C/4×1×1
    u = FC_uncertainty(s)        # B×1×1×1 (оцінка невизначеності)
    w = FC_expand(s) + u*noise     # B×C×1×1 (ваги з урахуванням
невизначеності)
    return x × w, u
```

Оцінка невизначеності поширюється через усі шари мережі, накопичуючи пошарову невизначеність через формулу:

$$u\_total = 1 - \Pi\_l (1 - u\_l)$$

Така мультиплікативна агрегація забезпечує природне зростання невизначеності при проходженні через глибші шари мережі, відображаючи накопичення невизначеності через послідовні трансформації ознак.

EfficientNet-Lite для Raspberry Pi. EfficientNet-Lite оптимізується спеціально для платформ на базі ARM через серію архітектурних модифікацій. Злиті згортки об'єднують згортку, пакетну нормалізацію та активацію в єдину операцію, що значно знижує трафік пам'яті та покращує використання кешу. Симетричне заповнення замінює асиметричне заповнення оригінального EfficientNet для кращої сумісності з векторними інструкціями ARM NEON. Вивід з фіксованою роздільною здатністю відмовляється від динамічного масштабування роздільної здатності на користь передбачуваної затримки, що критично для застосувань реального часу.

Складене масштабування адаптується для компромісу невизначеність-затримка через розширену формулу:

$$depth: d = \alpha^{\varphi}$$
$$width: w = \beta^{\varphi}$$
$$resolution: r = \gamma^{\varphi}$$
$$uncertainty\_capacity: u = \delta^{\varphi}$$

Складений коефіцієнт φ контролює загальний масштаб моделі, а новий фактор масштабування δ дозволяє незалежно керувати ємністю для оцінки невизначеності. Це дає можливість створювати моделі з різним балансом між загальною точністю та якістю оцінки невизначеності залежно від вимог конкретного застосування.

ShuffleNet з апроксимацією Монте-Карло Dropout. Архітектура ShuffleNet природно підходить для ефективної апроксимації вибірки Монте-Карло завдяки своїй операції перемішування каналів, яка створює варіації підмереж:

```
def ShuffleBlock_MC(x):

# Розділення каналів
x1, x2 = ChannelSplit(x)

# Гілка 1: ідентичність з dropout
y1 = SpatialDropout(x1, p=0.1)

# Гілка 2: вузьке місце
y2 = Conv1x1(x2)
y2 = ChannelShuffle(y2)
y2 = DepthwiseConv3x3(y2)
y2 = Conv1x1(y2)

# Невизначеність через дисперсію
out = Concat([y1, y2])
var = Var([y1, y2])
return out, var
```

Оцінка невизначеності відбувається через обчислення дисперсії між гілками після їх конкатенації. Така архітектура дозволяє отримувати оцінки невизначеності з мінімальним навантаженням, оскільки різні гілки вже присутні в базовій архітектурі ShuffleNet. Структурований dropout між операціями перемішування створює корельо-

вані збурення, які краще апроксимують справжню невизначеність моделі порівняно з традиційним dropout.

Паралелізація з урахуванням обладнання. Периферійні блоки NPU та DSP сучасних мобільних процесорів вимагають спеціальних стратегій для ефективного використання їх обчислювальних можливостей. Пакетна невизначеність обробляє кілька вирізок одного зображення одночасно для оцінки дисперсії, використовуючи можливості SIMD сучасних процесорів. Конвеєрний паралелізм дозволяє перекривати обчислення різних гілок невизначеності, мінімізуючи час простою обчислювальних блоків. Змішування точності використовує FP16 для обчислень магістралі, INT8 для головок класифікації та FP32 для критичних операцій калібрування, оптимально розподіляючи навантаження між різними виконавчими блоками.

На прикладі Snapdragon 8 Gen 2 така комплексна оптимізація забезпечує трикратне прискорення порівняно з послідовною обробкою. Це досягається через одночасне використання Hexagon DSP для операцій INT8, Adreno GPU для обчислень магістралі FP16 та Kryo CPU для координації та калібрування FP32. Розподілене виконання також покращує теплові характеристики, дозволяючи підтримувати стійку продуктивність протягом тривалих сесій виводу.

Оптимізація для мобільних платформ. Смартфони (iOS/Android). Для Apple Neural Engine оптимізація включає специфічні адаптації під фреймворк Core ML. Квантизація ваг до INT8 використовує поканальні масштаби для збереження точності при зменшенні відбитка пам'яті. Квантизація активацій застосовує динамічні діапазони, адаптуючись до статистики конкретного входу. Критично важливим є збереження головок невизначеності в точності FP16, оскільки вони чутливіші до помилок квантизації:

```
let uncertaintyKernel =
MPSCNNFullyConnected(...)
uncertaintyKernel.destinationFeatureChannels = 1
uncertaintyKernel.neuronType = .sigmoid
```

Стратегія адаптивного виводу динамічно обирає режим роботи залежно від системних умов. В режимі низького енергоспоживання використовується одна модель з простим температурним масштабуванням для мінімізації енергоспоживання. Нормальний режим активує допоміжні головки невизначеності для кращої якості оцінки невизначеності без значного навантаження. Режим продуктивності запускає міні-ансамбль з 2 моделей для максимальної якості оцінки невизначеності, коли доступні достатні обчислювальні ресурси та заряд батареї.

Платформа Android з TensorFlow Lite використовує делегування NNAPI для автоматичного розподілу обчислень між Hexagon DSP та GPU залежно від доступного обладнання. Динамічна квантизація селективно застосовує INT8 для певних шарів на основі можливостей обладнання та аналізу чутливості. Гнучкі делегати забезпечують безшовне відкочування до CPU для операцій, які не підтримуються спеціалізованим обладнанням, гарантуючи сумісність з широким спектром пристроїв.

Оптимізація Raspberry Pi. Платформи Raspberry Pi 4/5 вимагають особливої уваги до управління пам'яттю через обмежені ресурси. Завантаження моделі відбувається по частинах для уникнення підкачки, яка критично сповільнює вивід. Спільна пам'ять між процесами через mmap дозволяє ефективно використовувати обмежену оперативну пам'ять при багатопроцесному розгортанні. Кільцевий буфер для пакетної обробки мінімізує виділення пам'яті під час виконання, знижуючи навантаження від збирання сміття.

Векторизація ARM NEON забезпечує значне прискорення критичних операцій:

```
// Векторизоване температурне масштабування
float32x4_t scale = vdupq_n_f32(1.0f / temperature);
for(int i = 0; i < n; i += 4){
float32x4_t logits = vld1q_f32(&input[i]);
float32x4_t scaled = vmulq_f32(logits, scale);
vst1q_f32(&output[i], scaled);
}
```

Багатопотокова стратегія використовує пул потоків з 4 робочими, відповідно до числа ядер у Raspberry Pi 4, з пакетним паралелізмом для агрегації невизначеності та асинхронним вводом-виводом для попередньої обробки зображень, максимізуючи пропускну здатність при збереженні низької затримки.

Енергоефективність. Оптимізація енергоспоживання стає критичною для пристроїв з батарейним живленням, де кожен міліджоуль має значення. Динамічне масштабування напруги/частоти адаптує робочу точку процесора залежно від навантаження, знижуючи частоту для некритичного виводу та застосовуючи прискорення для вимог реального часу. Прогнозування теплового дроселювання попереджує деградацію продуктивності через перегрів, проактивно знижуючи

навантаження перед досягненням критичних температур.

Стратегії вибіркових обчислень значно знижують середнє енергоспоживання. Механізми раннього виходу дозволяють припинити вивід для прогнозів з високою впевненістю без проходження через усі шари мережі. Каскадні моделі застосовують послідовність швидка → точна → невизначеність моделей, де кожна наступна активується лише при недостатній впевненості попередньої. Обробка на основі регіонів для великих зображень обробляє лише релевантні регіони замість повного зображення, економлячи обчислювальні ресурси.

Квантизація має драматичний вплив на енергоефективність. Операції INT8 споживають приблизно в 4 рази менше енергії на операцію порівняно з FP32, тоді як INT4 може досягти 8-кратного зниження, хоча потребує ретельного перенавчання для збереження точності. Підходи змішаної точності забезпечують оптимальний компроміс, використовуючи різну точність для різних частин мережі на основі аналізу чутливості.

Вимірювання на Pixel 6 Pro демонструють конкретні показники енергоспоживання. Базова модель FP32 споживає 850 мДж на зображення, що швидко виснажує батарею при безперервному використанні. Квантизована до INT8 версія знижує споживання до 210 мДж/зображення, забезпечуючи 4-кратне покращення. Комбінація INT8 з 30% обрізкою далі знижує споживання до 145 мДж/зображення. Температурне масштабування додає лише 2 мДж/зображення навантаження, що є незначним порівняно з загальною економією від оптимізацій.

Компроміси між точністю та ефективністю. Парето-оптимальні конфігурації. Різні застосування вимагають різного балансу між точністю, калібруванням та затримкою, формуючи границю Парето оптимальних конфігурацій. Режим наднизької затримки з вимогою менше 10 мс досягається через MobileNetV3-Small з простим температурним масштабуванням, забезпечуючи ECE 5.2% та точність 67.5% при затримці лише 8 мс. Така конфігурація ідеально підходить для застосувань відстеження в реальному часі, де швидкість критичніша за абсолютну точність.

Збалансована конфігурація в діапазоні 20-50 мс використовує EfficientNet-Lite1 з допоміжною головкою, досягаючи значно кращих метрик з ECE 3.1% та точністю 75.3% при затримці 28 мс. Це представляє оптимальний вибір для більшості практичних застосувань, де потрібен хороший баланс між якістю та швидкістю. Високоякісний режим для критичних рішень застосовує ансамбль з 2 моделей зі спільними вагами, досягаючи відмінної ECE 1.8% та точності 78.9%, хоча з вищою затримкою 72 мс, що все ще прийнятно для багатьох сценаріїв, критичних для безпеки.

Адаптивні стратегії. Адаптація під час виконання на основі контексту дозволяє динамічно обирати оптимальну стратегію виводу:

```
def adaptive_inference(image, battery_level, latency_budget):
    if battery_level < 20:
        # Мінімальний режим
        return fast_model(image), None
    elif latency_budget < 30:
        # Збалансований режим
        pred = balanced_model(image)
        unc = temperature_scale(pred, T_calibrated)
        return pred, unc
    else:
        # Повна невизначеність
        preds = [model(image) for model in ensemble[:2]]
        pred = np.mean(preds, axis=0)
        unc = np.var(preds, axis=0)
        return pred, unc
```

Каскадний вивід. Ієрархічна обробка оптимізує середню затримку через інтелектуальну маршрутизацію. Крихітна модель на першому етапі обробляє легкі зразки за 3 мс, фільтруючи приблизно 70% випадків з високим порогом впевненості понад 0.95. Середня модель на другому етапі обробляє випадки помірної складності, що складають близько 25% зразків, додаючи оцінку невизначеності за 20 мс. Повний ансамбль активується лише для справді неоднозначних випадків, які складають лише 5% від загальної кількості, витрачаючи 80 мс на повну квантифікацію невизначеності. Така стратегія забезпечує середню затримку всього 11.1 мс, що значно нижче найгіршого сценарію.

Практичні рекомендації. Вибір методу за сценарієм використання. Медичні додатки на смартфонах вимагають особливої уваги до надійності та інтерпретованості. Дистиляція знань від серверного ансамблю в поєднанні з навченою впевненістю забезпечує передачу високоякісних оцінок невизначеності до мобільної моделі. EfficientNet-Lite2 з допоміжною головкою невизначеності надає достатню ємність для складних медичних зображень. Квантизація INT8 застосовується вибірково зі збереженням критичних шарів у FP16 для збереження діагностичної якості. Розгортання через TFLite з делегуванням NNAPI на Android та CoreML на iOS забезпечує оптимальне використання обладнання. Очікувані метрики включають

ECE менше 3% для калібрування клінічного рівня, затримку під 50 мс для чутливого користувацького досвіду та час роботи батареї близько 2 годин безперервного використання.

Дрони та автономні системи потребують надійних оцінок невизначеності в динамічних середовищах. Легковагова апроксимація Монте-Карло Dropout через структурований dropout забезпечує практичну квантифікацію невизначеності без надмірних обчислювальних витрат. MobileNetV3-Large з 3 зразками dropout надає хороший баланс між якістю та ефективністю. Обрізка видаляє 30% некритичних каналів для зниження обчислювального навантаження. NVIDIA Jetson Nano з оптимізацією TensorRT забезпечує ефективне виконання на вбудованому GPU. Система досягає ECE менше 4%, FPS понад 15 для плавної роботи та споживання енергії під 5 Вт для подовженого часу польоту.

Камери спостереження IoT оптимізуються для безперервної цілодобової роботи з мінімальними ресурсами. Одна модель з температурним масштабуванням забезпечує базову обізнаність про невизначеність без значного навантаження. ShuffleNetV2 0.5x надає екстремальну ефективність при прийнятній точності для задач спостереження. Агресивна квантизація INT8 максимізує пропускну здатність на обмеженому обладнанні. Raspberry Pi 4 з оптимізаціями, специфічними для ARM, забезпечує економічно ефективне розгортання. Метрики включають ECE до 6%, що прийнятно для некритичного моніторингу, затримку під 100 мс для розумної чутливості та стабільну цілодобову роботу без перегріву чи деградації продуктивності.

Етапи впровадження. Профілювання цільової платформи є критичним першим кроком:

```
profile = benchmark_platform(
    models=[baseline, pruned, quantized],
    metrics=['latency', 'memory', 'power'],
    test_samples=1000)
```

Вибір базової архітектури вимагає ретельного аналізу границі Парето в просторі точність проти ефективність. Можливості апаратного прискорення визначають, які операції будуть ефективними, впливаючи на архітектурні вибори. Обмеження пам'яті часто стають лімітуючим фактором, особливо на вбудованих платформах, вимагаючи креативних рішень для стиснення моделі.

Конвеєр навчання починається з початкового навчання на потужній серверній інфраструктурі з повним набором даних для максимальної якості. Дистиляція знань поступово передає знання до моделі цільового розміру через багатоетапний процес. Точне налаштування з урахуванням квантизації адаптує модель до зниженої точності, мінімізуючи деградацію точності. Фінальне калібрування на специфічному для платформи валідаційному наборі забезпечує оптимальну продуктивність на фактичному обладнанні розгортання.

Оптимізація розгортання включає конверсію моделі до цільового формату через проміжне представлення ONNX. Специфічні для обладнання оптимізації використовують унікальні особливості кожної платформи для максимальної ефективності. Налаштування параметрів виконання оптимізує розподіл потоків, буфери пам'яті та політики планування. A/B тестування різних конфігурацій у виробничому середовищі підтверджує метрики продуктивності та якості.

Моніторинг та обслуговування. Виробничий моніторинг для периферійних моделей вимагає комплексних систем відстеження. Моніторинг статистики невизначеності відстежує розподіл оцінок впевненості для виявлення аномалій або дрейфу. Виявлення дрейфу ECE ідентифікує, коли калібрування моделі погіршується з часом, сигналізуючи про необхідність перекалібрування. Відстеження частоти зразків поза розподілом виявляє зміни у вхідному розподілі, що можуть вимагати оновлень моделі.

Збір метрик продуктивності включає детальні процентилі затримки, не лише середні значення, для розуміння поведінки хвоста. Шаблони споживання батареї виявляють неефективності або неочікувані споживання енергії. Події теплового дроселювання вказують, коли відбувається деградація продуктивності через перегрів, припускаючи необхідність покращення теплового управління.

Безперервне вдосконалення забезпечується через механізми федеративного навчання, які дозволяють оновлення моделі зі збереженням приватності від розподілених периферійних пристроїв. Співпраця периферія-хмара дозволяє вибіркове розвантаження складних випадків до хмари для детального аналізу. Автоматична адаптація гіперпараметрів налаштовує параметри виконання на основі спостережуваних шаблонів продуктивності, безперервно оптимізуючи поведінку системи для змінних умов та вимог.

Висновки. Проведене дослідження методів швидкої оцінки невизначеності для edge computing у системах комп'ютерного зору дозволяє зробити наступні висновки. Knowledge distillation від teacher

ensembles до student моделей забезпечує найкращий компроміс між якістю калібрування та ефективністю, досягаючи 85-90% performance teacher при 10-кратному зменшенні витрат. Temperature scaling залишається незамінним baseline методом через нульові додаткові витрати та 50-70% покращення ECE. Auxiliary uncertainty heads додають мінімальний overhead (3-5%) при суттєвому покращенні калібрування для single-model deployment.

INT8 квантизація з calibration-aware training забезпечує 4x compression при ECE degradation < 1%, що критично для memory-constrained devices. Hardware-specific оптимізації (NPU acceleration, NEON vectorization) дають додаткове 2-3x прискорення. Гібридні підходи, що поєднують distillation, quantization та temperature scaling, досягають ECE < 3% при latency < 50ms на типових смартфонах.

Практична значимість полягає у формулюванні конкретних deployment стратегій для різних edge платформ та застосувань. Запропоновані методи готові до впровадження в системи мобільної медичної діагностики, автономної навігації дронів, інтелектуального відеоспостереження.

Подальші перспективи досліджень включають: розробку neuromorphic computing підходів для ultra-low power uncertainty; федеративне навчання для continuous calibration без централізованого збору даних; автоматизований пошук оптимальних архітектур через NAS з uncertainty-aware objectives; створення спеціалізованих hardware accelerators для probabilistic inference на edge.

Список літератури:

1. Tran D., Dusenberry M., van der Wilk M., Hafner D. Bayesian Layers: A Module for Neural Network Uncertainty. Advances in Neural Information Processing Systems. 2019. Т. 32. С. 14633-14645.

2. Malinin A., Mlodozeniec B., Gales M. Ensemble distribution distillation. International Conference on Learning Representations. Addis Ababa, 2020. 12 с.

3. Pearce T., Leibfried F., Brintrup A. Uncertainty in neural networks: Approximately Bayesian ensembling. International Conference on Artificial Intelligence and Statistics. Virtual, 2020. С. 234-244.

4. Howard A., Sandler M., Chen B., Wang W., Chen L. C., Tan M. та ін. Searching for MobileNetV3. International Conference on Computer Vision. Seoul, 2019. С. 1314–1324.

5. Tan M., Le Q. EfficientNet: Rethinking model scaling for convolutional neural networks. International Conference on Machine Learning. Long Beach, 2019. С. 6105–6114.

6. Ma N., Zhang X., Zheng H. T., Sun J. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. European Conference on Computer Vision. Munich, 2018. С. 116-131.

7. Sensoy M., Kaplan L., Cerutti F., Saleki M. Uncertainty-aware deep classifiers using generative models. AAAI Conference on Artificial Intelligence. 2020. Т. 34, № 04. С. 5620–5627.

8. Gast J., Roth S. Lightweight probabilistic deep networks. Conference on Computer Vision and Pattern Recognition. Salt Lake City, 2018. С. 3369–3378.

9. Van Amersfoort J., Smith L., Teh Y. W., Gal Y. Uncertainty estimation using a single deep deterministic neural network. International Conference on Machine Learning. Virtual, 2020. С. 9690–9700.

10. Louizos C., Reisser M., Blankevoort T., Gavves E., Welling M. Relaxed quantization for discretized neural networks. International Conference on Learning Representations. New Orleans, 2019. 15 с.

11. Havasi M., Peharz R., Hernández-Lobato J. M. Minimal random code learning: Getting bits back from compressed model parameters. International Conference on Learning Representations. New Orleans, 2019. 14 с.

12. Liu C., Chen L. C., Schroff F., Adam H., Hua W., Yuille A. L., Fei-Fei L. Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation. Conference on Computer Vision and Pattern Recognition. Long Beach, 2019. С. 82–92.

13. Guo Y., Yao A., Chen Y. Dynamic network surgery for efficient DNNs. Advances in Neural Information Processing Systems. 2016. Т. 29. С. 1379–1387.

Vinnychenko V.V. FAST UNCERTAINTY ESTIMATION METHODS FOR EDGE COMPUTING IN COMPUTER VISION SYSTEMS

The article is devoted to the study of fast uncertainty estimation methods for computer vision systems operating on edge devices with limited computational resources. Deploying calibrated deep learning models on mobile platforms creates a fundamental contradiction between the need for reliable confidence estimation and strict constraints on computational resources, memory, and power consumption. The problem becomes critical when creating real-time autonomous systems for mobile medical diagnostics, unmanned aerial vehicles, and intelligent surveillance cameras. Traditional uncertainty calibration methods such as Deep Ensembles and Monte Carlo Dropout are unsuitable for edge computing due to 5-20x increase in computational costs. The paper provides a systematic analysis of four main optimization approaches: uncertainty distillation using teacher-student architectures, single-shot methods based on auxiliary heads, quantization and pruning of

Bayesian networks, efficient ensembles with weight sharing. Special attention is paid to adapting methods for typical edge platforms – smartphones based on Snapdragon and Apple Bionic processors, as well as single-board computers Raspberry Pi 4/5. Lightweight architectures MobileNetV3, EfficientNet-Lite and ShuffleNet with modifications for uncertainty estimation are investigated. Theoretical analysis shows that knowledge distillation from full-size ensembles achieves 85-90% of teacher model calibration quality with 10x reduction in computational costs. Temperature scaling remains the most efficient baseline method with zero additional overhead during inference. A hybrid architecture is proposed that combines a distilled student model with learned temperature scaling and selective prediction for critical applications. Practical recommendations are formulated for various usage scenarios, taking into account the specifics of hardware acceleration on target platforms.

Key words: *edge computing, uncertainty calibration, mobile devices, knowledge distillation, quantization, lightweight architectures, computer vision.*

Дата надходження статті: 23.09.2025
Дата прийняття статті: 10.10.2025
Опубліковано: 16.12.2025