Универзитет у Београду
Електротехнички факултет

Реализација модела за класификацију слика хране уз коришћење трансферног учења

Дипломски рад

Ментор:
Др Бошко Николић

Кандидат:
Вук Алексијевић 2020/0599

Београд, Септембар 2024.

САДРЖАЈ

1. Увод

У данашње време, конволуционе неуронске мреже (енгл. *Convolutional Neural Network*) као и технике из области дубоког учења (енгл. *Deep Learning*) представљају револуционарну снагу у применама рачунарског вида (енгл. *Computer Vision*). Током протекле деценије, конволуционе неуронске мреже су донеле значајне промене у разним апликацијама, посебно у виду класификације слика.

Класификација слика подразумева процес додељивања једне или више категорија сликама, као што је предвиђање да ли се на слици налази неки предмет или не. Ипак, конволуционе неуронске мреже нису ограничене само на бинарну класификацију, већ се лако могу скалирати на хиљаде различитих класа, што је приказано кроз познати *ImageNet* скуп података (енгл. *Dataset*), који садржи хиљаде класа. Овај скуп податак се користи као стандард за мерење перформанси алгоритама у рачунарској визији.

Развој конволуционих неуронских мрежа и техника дубоког учења постао је доступан целој генерацији софтверских инжењера захваљујући јавно објављеним индустријским пакетима као што је *Tensorflow*. Овај алат омогућава употребу истих градивних блокова за стварање конволуционих неуронских мрежа које користи и компанија *Google*. Ови градивни блокови се користе за писање апликација за дубоко учење без потребе за ручним кодирањем операција за графичке процесоре (енгл. *Graphics Processing Unit*) или имплементацијом оптимизатора као што је стохастички градијентни спуст (енг. Stochastic gradient descent).

Једноставност употребе, захваљујући високим нивоима програмских интерфејса апликација (енгл. *Application Programming Interface*) као што су *Keras*, омогућава програмерима да брзо прототипирају комплексне графове дубоког учења.

У наредним поглављима овог рада, детаљно ће бити описане методологије које су коришћене за развој оваквог модела. Упркос свим значајним напрецима, развој ефикасних модела конволуционих неуронских мрежа за специфичне задатке и даље представља значајан изазов. Овај рад представља истраживачки процес тренирања конволуционе неуронске мреже за класификацију слика у специфичном домену. Основни циљ овог истраживања је развој и оптимизација модела који може прецизно препознавати и класификовати слике из унапред дефинисаног скупа категорија.

Овај рад се бави имплементацијом модела за класификацију слика који користи *CNN*, прилагођен специфичним задацима класификације у домену препознавања хране. У циљу постизања што бољих резултата, примењен је приступ трансферног учења користећи претходно обучени *InceptionV3* модел. Овај приступ омогућава значајно убрзање процеса обуке и оптимизацију перформанси модела.

У раду су коришћени алати *TensorFlow* и *Keras*, који представљају апликационе интерфејсе за имплементацију модела и обраду великог скупа података. Рад ће представити кораке у развоју система, описати употребљен скуп података, и дати анализу перформанси модела кроз процес тренинга и валидације.

2. ПОЈМОВИ И ИСТОРИЈАТ

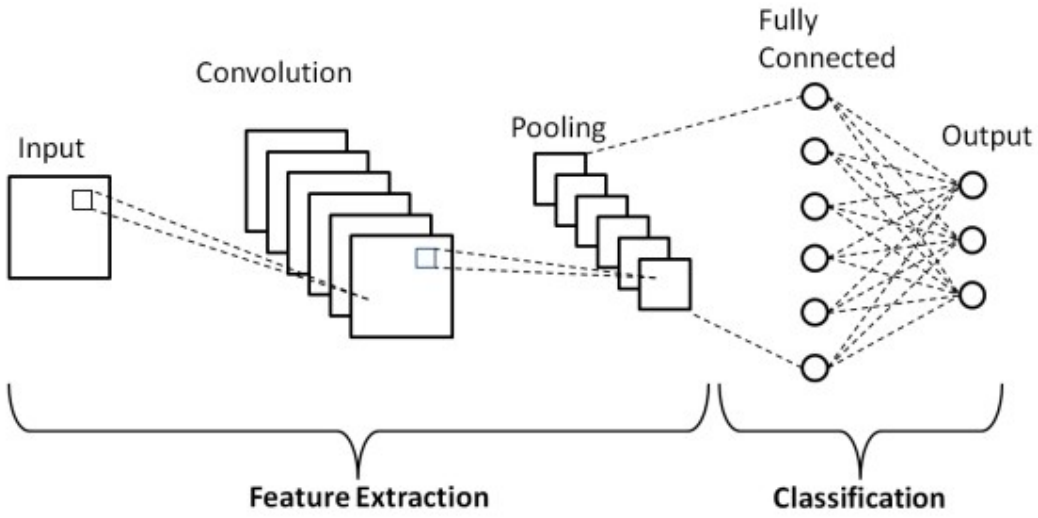
2.1. Структура конволуционих неуронских мрежа

Конволуционе неуронске мреже су специфична врста неуронске мреже које су посебно ефикасне за анализу података са обимном и комплексном структуром као што су слике. Овакав тип неуронских мрежа је посебно дизајниран да адаптивно уче хијерархију карактеристика из података. Будући да су за овакве примене специјализоване, оне су изузетни моћан алат за разне задатке из области рачунарске визије, попут класификације слика, детекције објеката, сегментације слике, итд.

Конволуционе неуронске мреже су сачињене од неколико слојева:

- Конволуциони слој: Овај слој представља основни слој конволуционих неуронских мрежа. Задатак овог слоја је да препознаје локалне карактеристике улазних података. На пример, у случају слика, конволуциони слој може да пропознаје ивице, облике, криве, затамљењна или текстуре. Овај слој користи филтере који се померају преко улазне слике и примењују операцију конволуције како би генерисали мапу карактеристика.
- Слој сажимања (енгл. *Pooling layer*): Овај слој је намењен за сажимање димензија података. Овим се постиже смањење броја параметара и рачунских сложености, док се изражавају најважније информације. Постоје две опције које се могу изабрати у оквиру овог слоја – сажимање максимумом (енгл. *Max pooling*) и сажимање помоћу средње вредности (енгл. *Average pooling*). Код сажимања максимумом се узима максимална вредност из прошлог слоја, и ова техника је најчешће коришћена. Са друге стране, сажимање помоћу средње вредности усредњава вредности и прошлог слоја.
- Потпуно повезан слој (енгл. *Fully connected layer*): Овај слој је сличан слојевима у традиционалним неуронским мрежема. Сви неурони су повезани са неуронима из претходног слоја. Овај слој уводи највећи ниво комплексности у модел, али је изузетно битан за коначан модел јер служи да се добијене карактеристике комбинују и класификују.
- Активациони слој (енг. *Activation layer*): У овом слоју се јавња нека функција активације која служи за увођење нелинеарности у модел. Ово омогућава да учи комплексне функције јер је сам проблем нелинеаран. Најпознатија активациона функција је *ReLU* (енгл. *Rectified Linear Unit*), која је коришћена у овом пројекту.

Сви релевантни слојеви и њихова повезаност је представљена на Слици 1.



Слика 1 – Слојеви конволуционе неуронске мреже.

Традиционалне методе препознавања слика се обично ослањају на ручно дефинисане карактеристике, што може бити нескалабилно и подложно грешкама. Конволуционе неуронске мреже имају способност да уче на релевантним карактеристикама улазних података током свог тренирања, чији је резултат боља и прецизнија анализа. Захваљујући конволуционим слојевима као и слојевима сажимања, овакве неуронске мреже су отпорне на било какав вид аугментације улазних података. Другим речима, транслација, ротација и остале варијације улазних података су ирелевантне за учење самог модела. Објекат на слици је препознат без обзира на његову позицију или оријентацију на слици. У поређењу са традиционалним моделима, конволуционе неуронске мреже захтевају мање предпроцесирања улазних података јер саме могу да науче и издвоје релевантне информације. Самим тим су и скалабилне на велике количине улазних података и могу да обраде слике са више канала захваљујући својој архитектури.

Због ових карактеристика, конволуционе неуронске мреже су постале један од најважнијих алата у обради вештачке интелигенције и машинског учења. Премда су посебно значајне када је реч о анализи визуелних података, ове не значи да нису применљиве на многе друге области као што су видео аналитика, препознавање говора, медицинска дијагностика, аутономна вожња, итд.

2.2. Историја и развој

Једна од првих инстанци неуронских мрежа се јавила 1980. године. Као претеча за саму област неуронских мрежа, Кунухико Фукушима (енгл. *Kunuhiko Fukushima*) је конструисао модел за екстракцију и препознавање писаних јапанских слова. Његов модел се звао Неокогнитрон (енгл. *Neocognitron*) и ослањао се на хијерархијску архитектуру са више слојева, пак није био трениран путем уназадне пропагације, већ је имао унапред дефинисане тежине.

Године 1998. се појавио један од првих успешних модела конволуционих неуронских мрежа. Њега су помогли да развију Јан Лекун (енгл. *Yann LeCun*), Леон Боту (енгл. *Leon Bottou*), Јошуа Бенђо (енгл. *Joshua Bengio*) и Патрик Хафнер (енгл. *Patrick Haffner*). Под именом *LeNet-5*, овај модел је био дизајниран за препознавање руком писаних цифара и био је кључан у аутоматској класификацији бројева на чековима. *LeNet-5* је користио конволуционе слојеве, слојеве зажимања и потпуно повезане слојеве, а трениран је путем уназадне пропагације грешке. Техника уназадне пропагације је после овог модела постала стандард за тренирање будућих конволуционих неуронских мрежа.

AlexNet је 2012. године представио прекретницу у развоју конволуционих неуронских мрежа. Овај модел су развили Алекс Крижевски (енгл. *Alex Krizhevsky*), Илија Сускевер (енгл. *Ilya Sutskever*) и Џефри Хинтон (енгл. *Geoffrey Hinton*), који су претходно поменуте године победили на такмичењу *ImageNet Large Scale Visual Recognition Challenge* (*ILSVRC*). Овај модел је имао значајне разлике у поређењу са претходним моделима на тражишту. Коришћење графичких процесора за тренирање неуронских мрежа, као и иновације као што су *ReLU* активациона функција и искључење (енг. *Dropout*), учинили су АлексНет моделом који је отворио пут за даљи развој дубоког учења.

VGGNet је 2014. године развијен од стране истраживача са Универзитета у Оксфорду. Овај модел је допринео једноставној и елегантној архитектури са дубоким слојевима, користећи мање филтере и повећавајући дубину мреже. Овај приступ је показао да повећавање дубине конволуционе неуронске мреже може значајно побољшати њене перформансе.

GoogLeNet, такође познат као *Inception*, био је још један кључан модел који је победио на *ILSVRC* 2014. године. Користио је иновативни приступ комбиновања више различитих филтера у истом слоју путем *Inception* модула, чиме је омогућио ефикасније рачунање и бољу употребу ресурса. 2015. године су истраживачи из фирме Microsoft развили модел под именом *ResNet*. Они су увели концепт "резидуалног учења", где слојеви уче само преосталу грешку (резидуал). Ово је омогућило изградњу веома дубоких мрежа (до 152 слоја) без проблема са деградацијом перформанси. *ResNet* је значајно побољшао перформансе у многим задацима рачунарског вида.

У последњих пола деценије се подрчје неуралних мрежа све више и брже развијало, где је најзначанији модел *EfficientNet* из 2019. године фокусиран на оптимизацију величине мреже кроз скалирање дубине, ширине и резолуције слика. Овај модел су развили истраживачи из фирме *Google*, и то је омогућило израду модела који су у исто време ефикасни и високо прецизни.

2.3. Циљеви рада

Један од кључних циљева је развој модела који могу постићи боље перформансе у задацима као што су класификација слика, детекција објеката, сегментација слика и слично. То укључује повећање тачности, смањење грешке, убрзање времена тренирања, и оптимизацију ресурса (меморије и процесорске снаге). Циљ је развити моделе који постижу високе перформансе уз мању сложеност, што значи мање слојева, мање параметара, и мању потрошњу ресурса. Ово је важно за примене на уређајима са ограниченим ресурсима, као што су мобилни уређаји или уградни системи.

Највећа примена оваквих модела је на задатке као што су препознавање лица, аутономна вожња, медицинска дијагностика путем анализа слика, надзор и сигурност, као и многе друге области где је анализа слика највећа удео посла. Ефикасно припремање и анотирање великих скупова података за обуку конволуционих неуронских мрежа, као и побољшање метода за тренирање модела је основа за модел, како би се постигла боља генерализација на невиђене податке.

Овакви модели су често "црне кутије" (енгл. *Black Box*), где је тешко разумети зашто модел доноси одређене одлуке. Један од циљева је развити методе које омогућавају боље разумевање унутрашњег функционисања модела и интерпретацију њихових одлука, што је посебно важно у областима као што су медицина или аутономна возила, где су људски животи у питању. Развој метода које омогућавају неуронским мрежама да препознају и рукују ситуацијама где су несигурни у своје одлуке, или да могу да идентификују и исправе грешке у предикцијама.

Циљ је развити моделе који су отпорни на различите врсте шума, варијације у подацима, и нападе. Ово осигурава да модел може поуздано радити у реалним условима. Модели конволуционе неуронске мреже често пате од проблема преносивости на нове, непознате домене (нпр. различите врсте слика или услови осветљења). Циљ је развити методе које омогућавају моделима да се боље генерализују на различите домене и услове.

3. ОСНОВЕ КОНВОЛУЦИОНИХ НЕУРОНСКИХ МРЕЖА

3.1. Архитектура

Архитектура конволуционе неуронске мреже је дизајнирана да ефикасно обрађује вишедимензионалне податке у облику слика или звучних сигнала. Свака конволуциона неуронска мрежа се састоји од различитих слојева који имају специфичне функције.

Улазни слој је први слој у конволуционој неуронској мрежи и представља тачку где се сирови подаци уносе у мрежу. У случају слика, подаци се представљају као тродимензионална матрица пиксела, где димензије укључују висину и ширину слике, као и број канала боја (на пример, три канала за RGB слике). Улазни слој је одговоран за правилан пријем и иницијално представљање података како би се могли даље обрађивати у каснијим слојевима. Улазне димензије обично укључују висину (H), ширину (W) и број канала (C). На пример, за слику величине 224x224 пиксела са три канала (RGB), димензије улазног слоја би биле 224x224x3. Поред тога, улазни слој може укључивати и нормализацију података како би се вредности пиксела скалирале у опсег погодан за даљу обраду.

Конволуциони слој је кључни део CNN-а који врши операцију конволуције над улазним подацима. Конволуција подразумева примену сетова филтера (такође познатих као језгра) на улазну слику, чиме се издвајају локалне карактеристике као што су ивице, текстуре и сложенији облици. Сваки филтер пролази кроз слику и израчунава тачкасти производ између филтера и дела слике, чиме се генерише мапа карактеристика (енг. feature map). Ове мапе карактеристика затим служе као улаз за следеће слојеве у мрежи.

Конволуциони слој има следеће параметре:

- Филтери (језгра): Мање матрице (нпр. 3x3, 5x5) који се померају преко улазне слике. Величина филтера је параметар који је унапред дефинисан и директно утиче на степен детаља који ће мрежа бити способна да ухвати.
- Количина филтера: Одређује број излазних канала или мапа карактеристика. Већи број филтера омогућава мрежи да учи више различитих карактеристика из слике. На пример, ако се користи 32 филтера у једном конволуционом слоју, излаз из тог слоја ће имати 32 канала, сваки са сопственом мапом карактеристика.
- Активирање: Након конволуције, обично се примењује функција активације као што је *ReLU* (енгл. *Rectified Linear Unit*) како би се додала нелинеарност у модел и омогућило мрежи да учи сложеније функције.

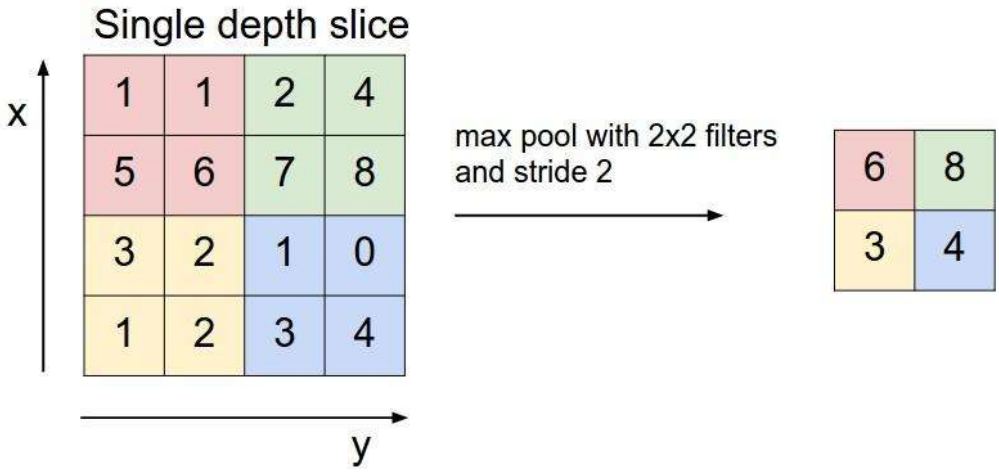
Слој активације је неопходан за увођење нелинеарности у модел, што омогућава мрежи да учи и представља сложене релације у подацима. У пракси се најчешће користи *ReLU* функција, која трансформише излазе из конволуционих слојева тако што све негативне вредности поставља на нулу. *ReLU* функција се дефинише у скупу позитивних реалних

бројева, што значи да ако је вредност улазног пиксела негативна, она постаје нула, а ако је позитивна, остаје непромењена. Ова функција уводи нонлинеарност која је неопходна за успешно учење дубоких неуронских мрежа, омогућавајући мрежи да учи компликоване обрасце у подацима. Осим *ReLU*-а, понекад се користе и друге функције активације, као што су *Leaky ReLU* или сигмоидна функција, у зависности од конкретне примене. Више о активационом функцијама ће бити објашњено у следећем поглављу.

Слој сажимања служи за смањење димензионалности мапа карактеристика које производе конволуциони слојеви. Смањењем димензија, овај слој помаже у смањењу броја параметара у мрежи, што доводи до смањеног ризика од прекомерног учења (енгл. *overfitting*) и побољшава ефикасност обуке. Осим тога, слојеви сажимања могу да уоче најважније карактеристике у мапама карактеристика, што помаже у бољој генерализацији модела. Смањивањем димензија мапе карактеристика придодаје мањој рачунарској комплексности и мањем броју параметара које треба научити у каснијим слојевима мреже, али истовремено задржава најважније информације за класификацију.

Постоје две врсте слоје сажимања, које укључују:

- *Max Pooling*: У овој врсти pooling-а, узима се максимална вредност из сваког прозора (нпр. 2x2) унутар мапе карактеристика. Ова техника омогућава моделу да задржи најважније информације и игнорише мање значајне детаље. Пример ове врсте сажимања се налази на Слици 2.
- *Average Pooling*: Овде се израчунава просечна вредност унутар прозора. Ова метода задржава општу информацију о карактеристикама, али без наглашавања најјачих сигнала као код Max Pooling-а.



Слика 2 – Пример *Max Pooling* сажимања.

Потпуно повезани слој се налази на крају мреже и игра кључну улогу у класификацији. Сваки неурон у овом слоју је повезан са свим неуронима из претходног слоја, што омогућава комбинацију и интеграцију свих карактеристика које су научене у претходним слојевима. Овај слој у суштини претвара простор карактеристика у простор одлука, где свака могућа класа има своју одговарајућу вероватноћу. Потпуно повезани слој обрађује и комбинује карактеристике које су научене у претходним слојевима и генерише финалне излазе, као што су вероватноће класа. На пример, ако је задатак да се класификује слика у једну од десет класа, излаз из последњег потпуно повезаног слоја биће вектор дужине 10, где свака компонента представља вероватноћу да слика припада одређеној класи. Излази из претходног слоја (најчешће након примене технике *flattening*, где се вишедимензионална структура мапе карактеристика претвара у један вектор) улазе у потпуно повезани слој. Свака веза између неурона има своју тежину. Активација се израчунава множењем улазних вредности са тежинама, а затим се на то примењује активацијска функција (нпр. *sigmoid*, *softmax* или *ReLU*).

Наведени слојеви раде заједно како би се научиле релевантне карактеристике из свих карактеристика сирових података. Свака од ових компоненти доприноси укупној способности мреже да препозна и разликује сложене обрасце у подацима.

3.2. Активационе функције

Активационе функције играју битну улогу у раду неуронских мрежа, нарочито када су у питању конволуционе неуронске мреже. Њихова основна улога је да уведу нелинеарност у мрежу, што омогућава моделима да уче и моделирају сложене обрасце и односе у подацима.

Без активационих функција, сви слојеви мреже били би линеарни, што значи да би цела мрежа, без обзира на број слојева, могла моделирати само линеарне односе. Активационе функције омогућавају мрежи да учи нелинеарне функције, што је неопходно за решавање сложених проблема као што су препознавање слика, говор, и природни језик. Активационе функције омогућавају мрежама да моделирају сложене карактеристике као што су ивице, облици и текстуре, које су кључне за препознавање објеката на сликама.

Најчешће коришћене линеарне функције су следеће:

- *ReLU* (енгл. *Rectified Linear Unit*) је једна од најпопуларнијих активационих функција у конволуционим неуронским мрежама. Ако је улазна вредност позитивна, излаз је једнак улазу. Ако је улазна вредност негативна, излаз је нула. Оваква функција је брза за израчунавање и помаже у решавању проблема са експлодирајућим и нестајућим градијентима, који су чести у дубоким мрежама. Једноставна је за имплементацију, убрзава обуку, смањује проблем нестајућих градијената. Једна мана ове активационе функције је ‘умирање’неурона где неки неурони постану неактивни и више не доприносе учењу ако константно примају негативне вредности.
    - $f(x) = max(0, x)$

- *Leaky ReLU* је варијација *ReLU* функције која допушта мале негативне вредности, чиме се ублажава проблем ‘умирања’ неурона. Пружа сличне предности као *ReLU*, али са смањеним ризиком од ‘умирања’ неурона, али зато потенцијално може бити теже за оптимизацију у односу на стандардни *ReLU*.
    - $f(x) = max(0.01x, x)$

- Сигмоид је функција која на излаз доводи вредност између нуле и јединице. Ова активациона функција се често користи за бинарну класификацију и добра је за моделирање вероватноће. Ова функција има своје недостатке као што су проблем нестајућег градијента, што отежава учење у дубоким мрежама. Такође, вредности могу брзо ући у засићење (приближити нули или јединици), смањујући ефикасност учења.
    - $f(x) = \frac{1}{1 + e^{-x}}$

- Хиперболички тангенс је сличан сигмоидној функцији, али је подручје излаза од негативне јединице до позитивне јединице. Ова активациона функција је симетрична око нуле, што је чини погоднијом од сигмоидне функције у неким случајевима. Предност хиперболичког тангенса лежи у томе да је функција погодна за моделирање података који имају и позитивне и негативне вредности, али као и сигмоид, постоји проблем нестајућег градијента.
    - $f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$

- *Softmax* је функција претвара низ вредности у низ вероватноћа чија је сума један. Ова активациона функција се најчешће користи у последњем слоју неуронских мрежа за вишекласну класификацију. Корисна за проблеме где је потребно моделирати вероватноће више класа, али може бити рачунарски захтевна у односу на једноставне функције попут *ReLU*. Ова функција је коришћења за класификацију слика у овом раду.
    - $f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$

Нелинеарност уведена активационим функцијама повећава капацитет мреже да моделира сложене односе и омогућавају мрежама да уче нелинеарне обрасце, што је неопходно за успех неуралних мрежа у задацима као што су препознавање слика.

3.3. Процес учења

Процес учења код конволуционих неуралних мрежа се ослања на две главне компоненте: уназадно ширење грешке (енг. *backpropagation*) и оптимизацију. Процес учења мреже утиче на њихову способност да препознају и класификују слике.

Конволуционе неуронске мреже уче на основу великих скупова података кроз итеративни процес. Циљ је минимизација разлика између предикција мреже и стварних ознака (нпр. категорија слике) прилагођавањем тежина у мрежи. Овај процес укључује унапредно ширење и губитак. Унапредно ширење (енгл. *Forward Propagation*) подразумева улаз (нпр. слика) који пролази кроз мрежу, слој по слој, све до излазног слоја, где мрежа генерише предикцију. Губитак (енг. *Loss*) је функција губитка која израчунава разлику између стварне ознаке и предикције, односно претпоставке мреже. На пример, за класификацију може се користити *Cross-Entropy* функција губитка.

Уназадно ширење грешке је метода која се користи за подешавање тежина мреже на основу израчунатог губитка. Ова метода се састоје из два корака: израчунавање градијента и ажурирање тежина.

Градијент представља смер и брзину промена које треба извршити на тежинама како би се смањио губитак. Након што се израчуна губитак, уназадно ширење грешке почиње поново од излазног слоја мреже, и враћа се уназад кроз сваки слој, рачунајући градијент губитка у односу на сваку тежину. Када се градијенти израчунају, тежине у мрежи се ажурирају тако да се губитак смањи. Ажурирање се врши помоћу оптимизационе технике, обично градијентног спуштања (енг. *Gradient Descent*). Градијентно спуштање је метода оптимизације која користи градијенте за подешавање тежина. Формула за ажурирање тежине је:

$$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$

У овој формули, w је тежина, η је стопа учења, а $\frac{\partial L}{\partial \omega}$ је градијент губитка у односу на тежину.

Оптимизација је процес проналажења најбољих вредности тежина које минимизирају функцију губитка. Различите технике оптимизације могу се користити за ефикасније учење мреже:

- Стохастички градијентни спуст (енг. *Stochastic Gradient Descent*) је метода где се градијенти израчунавају и тежине се ажурирају након сваке појединачне инстанце података. Ова метода је брза, али може имати ’бучне’ путање учења.
- *Momentum* је техника која помаже у убрзању стохастичког градијентног спуста у правцу смера градијената, додајући део претходног градијента тренутном ажурирању. Ово помаже у избегавању осцилација и убрзава конвергенцију.
- *RMSprop* (енгл. *Root Mean Square Propagation*) се користи покретни просек квадрата претходних градијената за скалирање тренутног градијента. Ово омогућава аутоматско подешавање стопе учења за сваки параметар, чиме се стабилизује процес учења.

- *Adam* (*Adaptive Moment Estimation*) комбинује идеје из *momentum*-а и *RMSprop*-а. Ова метода одржава експоненцијално смањење средње вредности првог тренутка (средње вредности градијената) и другог тренутка (квадрата градијената). Адам је једна од најпопуларнијих метода за оптимизацију због своје ефикасности и прилагодљивости.

Поред оптимизација, постоје још параметара који помажу разумевању и развијању модела конволуционе неуронске мреже. Стопа учења η је хиперпараметар који одређује величину корака приликом ажурирања тежина. Ако је превелика, може узроковати нестабилно учење; ако је премала, може довести до спорог конвергирања. Регуларизација је техника која додаје пенале за комплексност модела, помажући да се избегне преобучавање (енгл. *overfitting*). Епоха је један пролаз кроз цео скуп података. Током сваке епохе, мрежа пролази кроз сваки узорак у скупу података и ажурира тежине. Рано заустављање (енг. *Early Stopping*) је техника за прекид обуке ако се перформансе на валидационом скупу не побољшавају након одређеног броја епоха, како би се избегло преобучавање.

4. ПРИМЕНА КОД КЛАСИФИКАЦИЈЕ СЛИКА

Класификација слика је један од основних проблема у области рачунарског вида и машинског учења, где је циљ аутоматски препознати и доделити етикету или категорију свакој улазној слици. Ова технологија има широку примену у разним областима, укључујући аутономна возила, медицинску дијагностику, надзор, препознавање лица и многе друге.

4.1. Основни концепт класификације слика

Класификација слика подразумева додељивање једне (или више) унапред дефинисаних етикета (класа) непознатој слици. На пример, ако систем за класификацију слика прими слику пса, требало би да је класификује као ’пас’. Улаз је слика у форми мреже пиксела (нпр. 224x224 пиксела, са три канала за боје: црвена, зелена, плава), док је излаз категорија или етикета која најбоље описује садржај слике (нпр. "пас", "мачка", "ауто").

Класификација слика је сложен проблем због неколико фактора, на пример висока димензионалност података. Слике су обично високо димензионални подаци. На пример, слика од 224x224 пиксела с три канала за боје има укупно 150,528 карактеристика (димензија). Овај велики број улаза захтева сложене моделе за обраду. Још један проблем је варијабилност слика. Исти објекат може изгледати различито у зависности од угла гледања и перспективе, као и осветљења и положаја. Неке слике могу бити скалиране и имати неку врсту шума (нпр. бити мутне или оштећене). Такође, неке класе су јако сличне, што отежава њихово распознавање. На пример, бурек са сиром и бурек са месом имају исти облик и могу се чинити истим на први поглед.

Битно је напоменути да постоје више модела за проблем препознавања објекта на слици, али је најуспешнији модел конволуционе неуронске мреже. Коришћењем конволуционих слојева, мрежа учи филтре који аутоматски препознају својства попут ивица, текстура и облика, што омогућава препознавање сложених образаца у сликама. Такође су јавно доступни претходно обучени модели као *VGGNet*, *ResNet*, и *Inception*. Овакви модели се користе као основни модел за подешавања (енгл. *Fine-tuning*) која су потребна за специфичне задатке класификације. У овом раду је коришћен основни модел *InceptionV3*.

4.2. Поступак класификације

Пре него што било који улазни податак уђе у модел, он мора бити предпроцесуиран. Слике се нормализују, скалирају и трансформишу како би биле компатибилне с моделом. Улазна слика пролази кроз неколико слојева мреже, укључујући конволуционе слојеве, слојеве сажимања и потпуно повезане слојеве. На крају мреже, последњи слој (обично потпуно повезани слој) даје низ вероватноћа за сваку класу. Класа са највећом вероватноћом постаје предикција мреже.

Перформансе модела за класификацију слика оцењују се помоћу различитих метрика. Једна од оваквих метрика је тачност (енг. Accuracy) и представља проценат исправно класификованих слика. Прецизност и осетљивост су метрике које се користе за дубље разумевање перформанси у задацима са неуравнотеженим класама. Још један занимљив преглед перформанси је матрица конфузије која чини табеларни приказ броја исправних и погрешних предикција за сваку класу, што помаже у анализи где модел греши.

4.3. Претпроцесирање слика

Најчешћи начин да се слика обради пре уласка у саму мрежу је нормализација. Нормализација је поступак који се користи за скалирање вредности пиксела унутар слика на опсег погодан за обраду од стране мреже, најчешће у опсегу од 0 до 1 или -1 до 1. Ово се постиже дељењем вредности сваког пиксела са максималном могућом вредношћу пиксела (на пример, 255 за 8-битне слике). Нормализација побољшава стабилност и брзину конвергенције током тренинга модела, јер смањује разлике у динамичком опсегу података.

При уласку у модел, потребно је да свака слика има неопходне димензије, што се постиже скалирањем. Скалирање подразумева промену димензија улазне слике тако да се подударају са улазним димензијама модела. На пример, ако модел очекује улазне слике димензија 224x224 пиксела, све улазне слике морају бити скалиране на те димензије.

Трансформације могу укључивати разне технике као што су ротација, премештање, рефлексија и изрезивање делова слике, а користе се за повећање варијабилности у скупу података. Ово помаже у смањењу ризика од прекомерног учења (енг. *overfitting*) и побољшава генерализацију модела.

4.4. Обрада кроз мрежу

Након претпроцесирања, улазна слика пролази кроз низ слојева унутар саме мреже, од којих сваки има специфичну улогу у екстракцији и обради карактеристика.

Први слојеви мреже су обично конволуциони слојеви који користе филтере за издвајање локалних карактеристика из слике, као што су ивице, текстуре и облици. Ови слојеви стварају мапе карактеристика које представљају различите аспекте улазне слике. Како се слика пролази кроз више конволуционих слојева, екстраховане карактеристике постају све сложеније, омогућавајући мрежи да препознаје и сложеније објекте и структуре.

Након конволуционих слојева, често следе слојеви сажимања (најчешће *Max Pooling*), који смањују димензије мапа карактеристика. Ово смањење димензија помаже у смањењу рачунарске комплексности и броја параметара, а истовремено задржава најважније информације. На тај начин, мрежа постаје отпорнија на позиционе варијације у објектима на слици.

На крају мреже налазе се један или више потпуно повезаних слојева. Ови слојеви узимају излазне карактеристике из конволуционих и сажимајућих слојева и комбинују их како би донели одлуку о коначној класификацији. Свакој могућој класи се додељује вероватноћа, а класа са највишом вероватноћом се бира као предикција мреже.

4.5. Оцењивање перформанси

Да би се проценило колико је модел успешан у класификацији слика, користе се различите метрике перформанси.

Тачност је најчешће коришћена метрика и представља проценат исправно класификованих слика у односу на укупан број слика у скупу података. Тачност даје добар укупни преглед успешности модела, али може бити недовољна у случајевима када су класе неуравнотежене.

Прецизност (енг. *Precision*) и осетљивост (енг. *Recall*) су метрике које пружају детаљнији увид у перформансе модела, посебно у случајевима неуравнотежених класа. Прецизност мери проценат тачно предвиђених позитивних примера у односу на све примере које је модел означио као позитивне, док осетљивост мери проценат тачно предвиђених позитивних примера у односу на укупан број позитивних примера у датом скупу.

Матрица конфузије је табеларни приказ који показује колико је пута модел тачно или погрешно класификовао сваку класу. У редовима матрице налазе се стварне класе, док се у колонама налазе предвиђене класе. Овај приказ омогућава лакшу идентификацију класа са којима модел има проблема, као и типове грешака које најчешће прави (нпр. мешање две сличне класе).

После добијања резултата, извршава се детаљна анализа перформанси модела. Ово укључује не само квантитативну анализу кроз горе наведене метрике, већ и квалитативну анализу. Квалитативна анализа може укључивати преглед појединачних слика које су погрешно класификоване како би се разумело где модел греши. Ова анализа може открити потенцијалне недостатке у дизајну модела или у скупу података.

5. АЛАТИ ЗА РАЗВОЈ НЕУРОНСКИХ МРЕЖА

5.1. Преглед алата

TensorFlow и *Keras* су два широко коришћена софтверска алата за развој и примену вештачке интелигенције, посебно дубоких неуронских мрежа као што су конволуционе неуронске мреже. *TensorFlow*, развијен од стране *Google*-а, је веома моћна платформа отвореног кода која омогућава истраживачима и инжењерима да креирају моделе машинског учења и дубоког учења. *TensorFlow* подржава сложене операције у облику графова тока података и пружа алате за тренинг, тестирање и оптимизацију модела, што га чини погодним за велике пројекте који захтевају обраду великих количина података и сложене архитектуре мрежа.

Keras, с друге стране, представља библиотеку високог нивоа која ради као интерфејс за *TensorFlow*. Његова основна предност је једноставност употребе и брза имплементација модела. *Keras* омогућава корисницима да брзо дефинишу, тренирају и тестирају неуронске мреже, без потребе да се брину о сложеним детаљима као што су управљање меморијом или оптимизација на нивоу система. Захваљујући *Keras*-у, развојни инжењери могу да се фокусирају на дизајн модела и експериментисање са различитим архитектурама, док се *TensorFlow* брине о перформансама и оптимизацији.

Keras омогућава брзо прототипирање, док *TensorFlow* нуди дубинску контролу и скалабилност, што је корисно за примене у индустрији и истраживању. Оба алата раде заједно како би омогућили рад на подацима у циљу машинског учења.

5.2. Апликациони програмски интерфејси

У окружењу дубоког учења, апликациони програмски интерфејси могу се класификовати у две основне категорије: нижег и вишег нивоа. Ове категорије разликују се по степену апстракције који пружају кориснику, као и по контроли коју корисник има над процесом развоја модела.

API-ји нижег нивоа пружају кориснику значајан степен контроле над свим аспектима развоја модела, укључујући и најситније детаље архитектуре, оптимизације и управљања подацима. Ови API-ји омогућавају програмерима да дефинишу прилагођене алгоритме и да експлицитно управљају корацима обраде унутар неуронских мрежа. Примери таквих алата укључују *TensorFlow* и *PyTorch*, где корисници могу директно управљати тензорима, креирати прилагођене слојеве и контролисати ток података кроз графове. Иако ово омогућава максималну флексибилност и оптимизацију, захтева дубоко познавање математике и машинског учења, што може бити изазовно за мање искусне кориснике.

API-ји вишег нивоа, као што је *Keras*, пружају виши степен апстракције, омогућавајући корисницима да брзо и лако дефинишу сложене моделе без потребе за управљањем детаљима ниског нивоа. Ови API-ји су дизајнирани тако да буду интуитивни и једноставни за употребу, омогућавајући корисницима да се фокусирају на избор архитектуре и подешавање хиперпараметара. *Keras*, као пример, омогућава дефинисање модела кроз слојеве које корисник лако може конфигурисати без бављења детаљима као што су иницијализација тежина или управљање меморијом.

Главна разлика између ових нивоа лежи у трговини између флексибилности и једноставности. Док API-ји нижег нивоа омогућавају већу флексибилност и прецизну контролу, они захтевају дубље разумевање основних принципа и више времена за развој. Насупрот томе, API-ји вишег нивоа омогућавају брже прототипирање и лакше прилагођавање модела, али по цену губитка неке флексибилности и контроле. За напредне кориснике и специфичне примене, API-ји нижег нивоа су незаменљиви, док су API-ји вишег нивоа идеални за брз развој и примену модела, посебно у индустријским апликацијама где време развоја и лакоћа употребе играју кључну улогу.

5.3. Унапред-тренирани модели

Када је у питању примена дубоког учења у различитим областима, употреба унапред тренираних модела представља значајну предност како у погледу времена тако и у погледу ресурса потребних за развој и имплементацију ефикасних решења. У овом раду коришћен је унапред тренирани модел, што је омогућило значајно побољшање укупног процеса развоја.

Прва и најочигледнија предност коришћења унапред тренираних модела је штедња времена и ресурса. Тренинг дубоких неуронских мрежа, посебно оних са великом количином параметара, захтева огромне рачунарске ресурсе и време. Ово укључује не само процес тренинга већ и проналажење оптималних хиперпараметара, што често подразумева дуготрајан и скуп процес експериментисања. Унапред тренирани модели, као што су они засновани на архитектурама као што су *ResNet*, *VGG* или *BERT*, већ су тренирани на великим скуповима података и могу се брзо прилагодити специфичним задацима коришћењем техника као што је фино подешавање (енгл. *fine-tuning*).

Друга предност је повећана прецизност и стабилност модела. Унапред тренирани модели обично су тренирани на огромним и разноврсним скуповима података, што им омогућава да ухвате комплексне обрасце и особине које би иначе било тешко научити од нуле. Овај аспект је посебно значајан у задацима као што су класификација слика, обрада природног језика и препознавање говора, где су подаци често високо димензионални и хетерогени. Коришћењем оваквих модела, чак и при ограниченим ресурсима и мањим скуповима података, могуће је постићи врхунске резултате.

Трећа предност је лакше прилагођавање различитим апликацијама. Унапред тренирани модели могу се користити као основа за развој решења у широком пољу апликација. На пример, модел трениран за класификацију слика може се лако прилагодити за детекцију објеката, сегментацију или чак за генеративне задатке. Ово је могуће захваљујући слојевитој природи дубоких мрежа, где се могу поново искористити дубоки слојеви који уче сложене и апстрактне особине података, док се последњи слојеви могу прилагодити специфичним захтевима задатка.

6. ПРЕГЛЕД ДЕТАЉА РЕШЕЊА

6.1. Опис скупа података

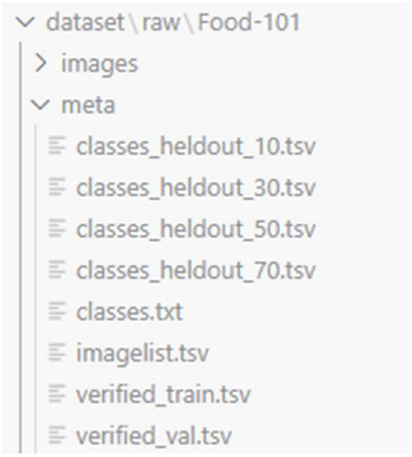
У овом раду коришћен је скуп података из рада *CleanNet: Transfer Learning for Scalable Image Classifier Training with Label Noise,* који представља важан ресурс у области машинског учења, посебно када је у питању класификација слика у условима присуства нетачних (енгл. *Noisy*) ознака.

Овај скуп података обухвата сликовне податке организоване у *JPEG* формату, са свим сликама прилагођеним тако да им је кратка ивица димензије 320 пиксела уз задржавање оригиналних димензија слике. Скуп класа садржи 101 класу, а листа тих класа је доступна у датотеци *meta/classes.txt*. Слике су расподељене у више скупова података, где сваки скуп има своје карактеристике и намену.

Скупови података су подељени у различите групе како би се омогућило темељно испитивање и валидација модела у контексту бучних ознака. Датотека *meta/imagelist.tsv* садржи листу свих слика заједно са одговарајућом класом. Осим тога, постоје и датотеке *meta/verified_train.tsv* и *meta/verified_val.tsv* које садрже слике са додељеним верификационим ознакама. Ове ознаке (*verification_label*) показују да ли је класа исправно додељена слици (1 – исправно, 0 – неисправно). Овај приступ омогућава процену ефикасности модела у контексту нетачних ознака и његову способност да учи из података упркос њиховој непоузданости.

Такође, у складу са циљем рада, скуп података укључује и листе класа које су посебно издвојене за истраживање утицаја трансферног учења на сузбијање буке у ознакама. Ове листе су дате у датотекама *meta/classes_heldout_10.txt*, *meta/classes_heldout_30.txt*, meta/classes_heldout_50.txt и *meta/classes_heldout_70.txt,* где су одређене класе изостављене из процеса верификације, омогућавајући да се тестира способност модела да преноси стечено знање из верификованих у неверификоване класе.

Визуелни приказ скупа података се налази на Слици 3.



Слика 3 – Скуп података.

6.2. Припрема података за тренинг

У овом раду, припрема података за тренинг извршена је коришћењем сета алата из библиотеке *TensorFlow* и *Keras*, са посебним нагласком на модул *ImageDataGenerator*, који омогућава обраду и подешавање слика.

Прво је учитана метаподатотека која садржи листу класа (*meta/classes.txt*) из скупа података. Свака класа је мапирана на свој индекс, што повезује слике са одговарајућим класама у каснијим фазама анализе. Као корак за припрему података користи се такође објекат класе *ImageDataGenerator*, који је подешен за аугментацију података кроз низ трансформација попут ротације, померања и хоризонталног/вертикалног окретања слика. Ове трансформације су коришћене да би се побољшала генерализација модела и његова способност да се носи са варијацијама у сликама током тренинга.

Ова подешавања и поставке су приказане на Слици 4.

```
with open('../../dataset/raw/Food-101/meta/classes.txt', 'r') as file:
    classes = [line.strip() for line in file.readlines()]
    class_to_index = dict(zip(classes, range(len(classes))))
    index_to_class = dict(zip(range(len(classes)), classes))
    class_to_index = {v: k for k, v in index_to_class.items()}

print(classes)

print("Setting up Image Generator...")
data_generator = ImageDataGenerator(
    featurewise_center = False,
    samplewise_center = False,
    featurewise_std_normalization = False,
    samplewise_std_normalization = False,
    zca_whitening = False,
    rotation_range = 45,
    width_shift_range = 0.125,
    height_shift_range = 0.125,
    horizontal_flip = True,
    vertical_flip = True,
    rescale = 1./255,
    fill_mode = 'nearest'
)
```

Слика 4 – Учитавање имена класа.

Затим су учитани подаци о сликама за тренинг и валидацију из датотека *verified_train.tsv* и *verified_val.tsv*, које садрже путање до слика и верификационе ознаке. Ове ознаке су коришћене да се из скупа података уклоне слике са нетачним ознакама, што је значајно смањило могућност да модел учи на погрешним подацима. Овај процес је приказан на Слици 5.

```
print("Loading training data into the Data Frame..")
train_df = pd.read_csv('../../dataset/raw/Food-101/meta/verified_train.tsv', sep='\t', header=None, names=['path', 'label'])
train_df = train_df[train_df['label'] == 1]  # Use only verified images
train_df['label'] = train_df['path'].apply(lambda x: x.split('/')[0])

print(train_df.head())  # Check that it has 'path' and 'label' columns

print("Loading validation data into the Data Frame..")
val_df = pd.read_csv('../../dataset/raw/Food-101/meta/verified_val.tsv', sep='\t', header=None, names=['path', 'label'])
val_df = val_df[val_df['label'] == 1]  # Use only verified images
val_df['label'] = val_df['path'].apply(lambda x: x.split('/')[0])

print(val_df.head())  # Check that it has 'path' and 'label' columns
```

Слика 5 – учитавање података о сликама.

Коначно, генерисани су тренинг и валидационих скупова уз помоћ метода *flow_from_dataframe* из *ImageDataGenerator* класе. Ова метода је омогућила аутоматско учитавање и аугментацију слика из тренинг и валидационих скупова, а такође је обезбедила да подаци буду коректно обележени и припремљени за тренинг процес. Конфигурација укључује параметре као што су величина слике (299x299 пиксела), величина партије података (енгл. *batch size*) и број класа.

```
print("Making training Data Generator from Data Frame for Training data")
train_generator = data_generator.flow_from_dataframe(
    train_df,
    directory=main_path,
    x_col='path',
    y_col='label',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    classes=classes
)

print("Making training Data Generator from Data Frame for Validation data")
val_generator = data_generator.flow_from_dataframe(
    val_df,
    directory=main_path,
    x_col='path',
    y_col='label',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    classes=classes
)
```

Слика 6 – Приказ генератора слика.

6.3. Имплементација модела користећи *Keras API*

У овом раду, имплементација модела коришћењем *Keras*-а извршена је преко комбинације претходно обученог модела и прилагођених слојева, чиме је обезбеђено да модел искористи предности трансферног учења уз додатно прилагођавање специфичним захтевима задатка класификације слика.

Први корак у имплементацији модела је учитавање претходно обученог модела *InceptionV3*, који је коришћен као основа за даљу надоградњу. Овај модел је иницијализован са тежинама обученим на скупу података *ImageNet*, а горњи (класификациони) слојеви су уклоњени. Такође, као улаз у модел дефинисан је тензор величине 299x299 пиксела са три канала, што одговара стандардним димензијама слика у овом пројекту.

На излаз овог претходно обученог модела додат је прилагођени низ слојева како би се омогућила класификација у специфичне категорије задатка. Прво је додат слој *GlobalAveragePooling2D*, који је изравнао просторне димензије излазних мапа карактеристика и претворио их у једнодимензионални тензор. Затим је додат густо повезан (*Dense*) слој са 4096 јединица, праћен нормализацијом путем *BatchNormalization* и активацијом са *ReLU* функцијом. Да би се смањила могућност преобучавања додат је и *Dropout* слој са стопом испуштања од 0.5. Истренирани модел који је коришћен у ради је приказан на Слици 7.

```
print("Clearing previous session ..")
kb.clear_session()

print("Defining the model..")
pre_trained_model = InceptionV3(weights='imagenet', include_top=False, input_tensor=Input(shape=(299,299,3)))
tensor = pre_trained_model.output
tensor = GlobalAveragePooling2D()(tensor)
tensor = Dense(4096)(tensor)
tensor = BatchNormalization()(tensor)
tensor = Activation('relu')(tensor)
tensor = Dropout(0.5)(tensor)

predictions = Dense(len(classes), activation='softmax')(tensor)

model = Model(inputs=pre_trained_model.input, outputs = predictions)
for layer in pre_trained_model.layers:
    layer.trainable = False

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])

steps_per_epoch = len(train_df) // batch_size
validation_steps = len(val_df) // batch_size
```

Слика 7 - *InceptionV3* модел.

Завршни корак у имплементацији је додавање излазног слоја са бројем неурона који одговара броју класа у задатку, уз употребу *softmax* активационе функције како би се обезбедила коректна вероватносна расподела излазних вредности.

Након дефинисања модела, слојеви претходно обученог *InceptionV3* модела су у почетној фази тренинга замрзнути (означени као нетренажни), чиме је омогућено да се нови слојеви адаптирају на задатак без ремећења претходно научених карактеристика. Компилација модела је извршена коришћењем *RMSprop* оптимизатора, који је изабран због његове ефикасности у управљању великим скуповима података и сложеним моделима, уз коришћење *categorical_crossentropy* функције губитка, која је стандард за мултикатегоријалну класификацију. Овај поступак, означен као први пролаз, је приказан на Слици 8.

```
print("First pass..")
checkpoint = ModelCheckpoint(filepath=saved_path_first, verbose=1, save_best_only=True)
csv_logger = CSVLogger('first.3.log')

history = model.fit(train_generator,
    validation_data=val_generator,
    validation_steps = validation_steps,
    steps_per_epoch=steps_per_epoch,
    epochs=10,
    verbose=1,
    callbacks=[csv_logger, checkpoint]
)
```

Слика 8 – Први пролаз модела.

У другој фази тренинга, део слојева основног модела је одмрзнут и дозвољено је њихово учење како би се побољшале перформансе. Ова фаза је урађена уз помоћ *Stochastic Gradient Descent* оптимизатора уз малу стопу учења и моментум. Други пролаз модела је приказан на Слици 9.

```
for layer in model.layers[:172]:
    layer.trainable = False
for layer in model.layers[172:]:
    layer.trainable = True

print("Second pass..")
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9), loss='categorical_crossentropy', metrics=['accuracy'])
checkpoint = ModelCheckpoint(filepath=saved_path_second, verbose=1, save_best_only=True)
csv_logger = CSVLogger('second.3.log')

history = model.fit(train_generator,
    validation_data=val_generator,
    validation_steps = validation_steps,
    steps_per_epoch=steps_per_epoch,
    epochs=100,
    verbose=1,
    callbacks=[csv_logger, checkpoint]
)
```

Слика 9 – Други пролаз модела.

6.4. Подешавање модела

У овом раду, фино подешавање је имплементирано у две фазе тренинга, омогућавајући моделирање које балансира између задржавања већ научених представљања и прилагођавања новим подацима.

У првој фази тренинга, приступ трансферног учења је примењен тако што су сви слојеви *InceptionV3* модела били замрзнути. На овај начин, само додати прилагођени слојеви су се тренирали, док су основни слојеви задржавали своје претходно научене тежине, обучене на скупу података ImageNet. Ова фаза служи за брзу конвергенцију модела на новом скупу података, јер је омогућила фокус на специјализацију нових слојева без ремећења већ постојећих карактеристика.

Након почетне фазе тренинга и постизања стабилности у губицима и прецизности, приступило се другој фази која се односи на фино подешавање. У овој фази, последњих 172 слоја *InceptionV3* модела је одмрзнуто, што значи да је дозвољено њихово учење.

Фино подешавање је извршено коришћењем *Stochastic Gradient Descent* оптимизатора, са веома ниском стопом учења (0.0001) и моментумом од 0.9. Ниска стопа учења је изабрана како би се спречило драстично мењање тежина и тиме нарушавање претходно научених карактеристика модела. Уместо тога, тежине су полако прилагођаване специфичностима новог скупа података, омогућавајући моделирање сложених представљања неопходних за успешну класификацију.

Кроз овај процес, модел је у стању да задржи генерализоване карактеристике научене из великог и разноврсног скупа података као што је *ImageNet*, док истовремено прилагођава свој поглед на податке специфичне за нови задатак.

6.5. Приказ резултата

У овом одељку дипломског рада објашњавамо серверски део система који је развијен коришћењем *Python*-а и *Flask*-а, оквира за веб апликације, за обраду слика и предвиђање категорија хране. Овај код омогућава клијентима да отпреме слике хране преко корисничког интерфејса, док сервер, у позадини, обрађује слику и враћа резултате предвиђања засноване на машинском учењу.

Функција *model_predict* је одговорна за предвиђање на основу отпремљене слике. Слика се учитава и мења њена величина на 299x299 пиксела, што је стандардна величина коју очекује *InceptionV3* модел који се користи. Затим се слика претвара у низ, и додаје се нова димензија како би одговарала формату који *Keras* модел очекује. Након тога, слика пролази кроз *preprocess_input* функцију како би се подаци нормализовали. Овај процес је приказан на Слици 10.

```
def model_predict(image_path, model):
    my_image = image.load_img(image_path, target_size=(299,299))
    array_image = image.img_to_array(my_image)
    array_image = np.expand_dims(array_image, axis=0)
    array_image = preprocess_input(array_image)

    predictions = model.predict(array_image)

    N = 5
    top_N_indices = predictions[0].argsort()[-N:][::-1]
    top_N_labels = []
    for i in top_N_indices:
        label = index_to_class[i].replace('_', ' ')
        confidence = predictions[0][i] * 100
        formatted_confidence = f"{confidence:.2f}%"
        top_N_labels.append((label, formatted_confidence))

    return top_N_labels
```

Слика 10 – Функција за класификацију слике.

На Слици 11 је дефинисана главна рута */predict*, која прима *POST* захтеве када корисник пошаље слику. Прво се проверава да ли је датотека присутна у захтеву и ако није, враћа се одговарајућа порука о грешци. Ако је датотека правилно отпремљена, чува се у привременом фолдеру, а затим се прослеђује функцији *model_predict* за добијање предвиђања. Након што се резултати добију, привремена датотека се брише како би се ослободио простор.

```python
@app.route('/predict', methods=['POST'])
def predict():
    if 'file' not in request.files:
        return jsonify({"error": "No file part"}), 400

    file = request.files['file']
    if file.filename == '':
        return jsonify({"error": "No selected file"}), 400

    if file:
        file_path = "./" + file.filename
        file.save(file_path)

        top_N_labels = model_predict(file_path, model)

        os.remove(file_path)

        return jsonify(top_N_labels)
```

Слика 11 – Обрада података преко захтева.

У оквиру овог дипломског рада, имплементиран је једноставни кориснички интерфејс за постављање слика хране и добијање предвиђања помоћу машинског учења. За ову сврху коришћен је *HTML* у комбинацији са *JavaScript*-ом ради креирања функционалности која омогућава корисницима да отпреме слику, покрену процес предвиђања и виде резултате директно у прегледачу, без потребе за освежавањем странице. У наставку је приказан фрагмент кода који омогућава ову интеракцију.

Основна сврха овог *HTML* документа јесте да омогући корисницима да отпреме слику хране, која ће потом бити прослеђена серверу ради предвиђања. Форма за отпремање дефинисана је елементом <*form*>. Елемент <*input*> омогућава кориснику да изабере слику са свог уређаја, док <*button*> иницира процес слања података. Код ове форме се налази на Слици 12.

```html
<h1>Upload a Food Image to Get Predictions</h1>

<form id="uploadForm" enctype="multipart/form-data">
    <input type="file" id="fileInput" name="file" accept="image/*" required><br><br>
    <button type="submit">Upload and Predict</button>
</form>

<div id="loadingIndicator">Loading... Please wait.</div>

<div id="predictions"></div>
```

Слика 12 – Форма *HTML* странице

JavaScript код у скрипти преузима улогу обраде овог догађаја и користи асинхрони *fetch* позив за комуникацију са сервером. На овај начин клијент комуницира са сервером без поновног освежавања странице.

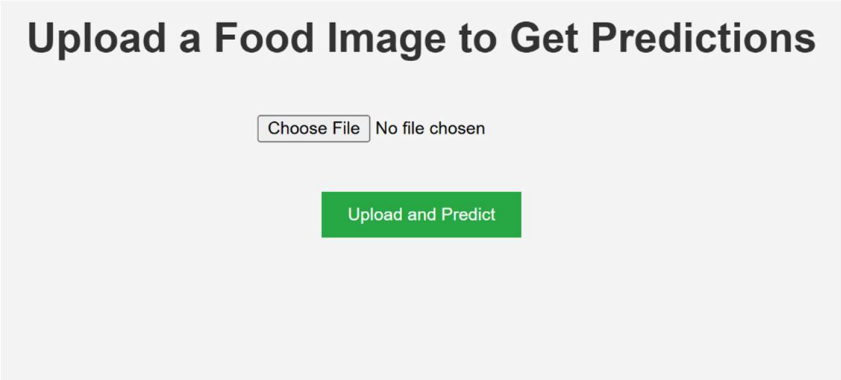
Функционалност ове апликације ослања се на асинхрони *fetch* метод који шаље отпремљену слику серверу у облику *FormData* објекта. Овај приступ омогућава серверу да прими слику као бинарни податак и процесира је помоћу одговарајућих алгоритама за машинско учење. Након што сервер врати резултате у *JSON* формату, функција *displayPredictions* приказује предвиђања у интерфејсу, динамички их додајући у *HTML* структуру странице.

Скрипта која врши асинхрони позив комуницирања са сервером се налази на Слици 13.

```
<script>
    document.getElementById('uploadForm').onsubmit = async function (event) {
        event.preventDefault();
        const loadingIndicator = document.getElementById('loadingIndicator');
        const predictionsDiv = document.getElementById('predictions');

        // Show loading indicator and clear previous predictions
        loadingIndicator.style.display = 'block';
        predictionsDiv.innerHTML = '';

        let formData = new FormData();
        formData.append("file", document.getElementById('fileInput').files[0]);

        const response = await fetch('/predict', {
            method: 'POST',
            body: formData
        });

        const result = await response.json();
        displayPredictions(result);

        // Hide the loading indicator after predictions are displayed
        loadingIndicator.style.display = 'none';
    };

    function displayPredictions(predictions) {
        const predictionsDiv = document.getElementById('predictions');
        predictionsDiv.innerHTML = '<h2>Predictions:</h2>';
        predictions.forEach(prediction => {
            predictionsDiv.innerHTML += `<p>${prediction[0]}: ${prediction[1]}</p>`;
        });
    }
</script>
```

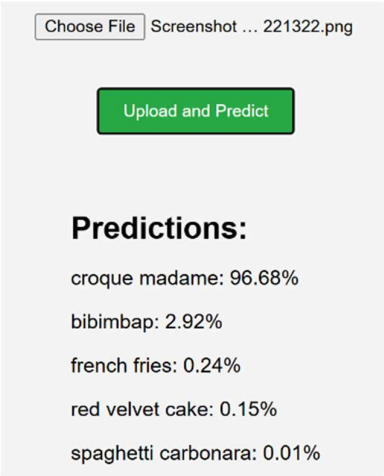
Слика 13- Асинхрони позив серверу

Када корисник отвори страницу, на ектрану је приказана стилизована форма са елементима где је омогућено слање слике на сервер. Када се изабере жењена слике и притисне дугме за слање, та слика ће бити обрађена од стране конволуционе неуронске мреже. Током чекања, корисинику је приказан индикатор да је обрада у току. Овако описана форми је приказана на Слици 14.



Слика 14 – Форма за унос слике

У тренутку када модел обради слику и избаци своје предикције, он тај резултат пошаље назад на страницу. Добијени резултати су скраћени на пет најрелевантнијих класа, где је за сваку класу приказана вероватноћа да унета слика припада одређеној класи. Резултат рада модела и сервера је приказан на Слици 15.



Слика 15 – Предвиђања модела

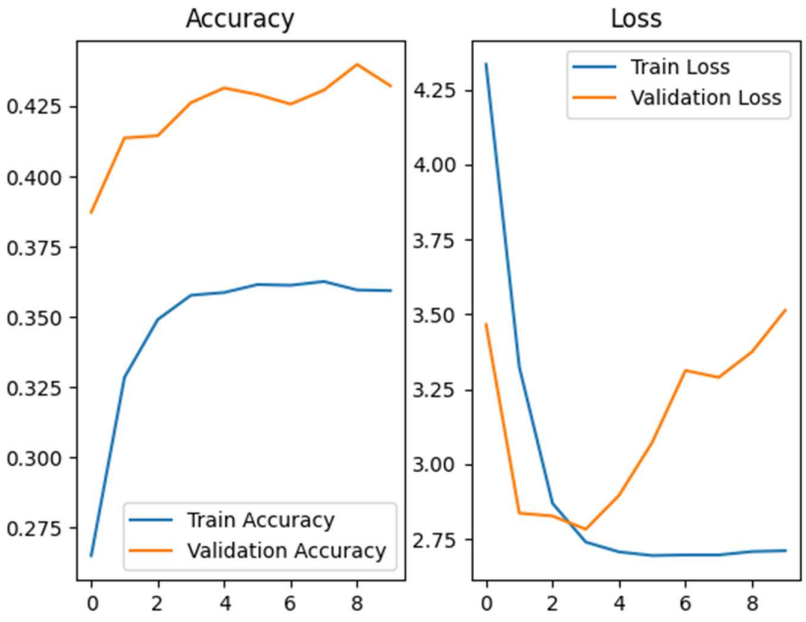
7. Резултати

Током тренинга модела за класификацију слика хране уз коришћење трансферног учења, праћени су резултати тачности (енгл. *accuracy*) и губитка (енгл. *loss*) на тренинг и валидационом скупу података. Ови резултати приказани су на сликама у наредним поглављима.

7.1. Тренинг и валидација у првом пролазу

На Слици 16, која представља почетних 10 епоха тренинга модела, примећује се значајно смањење губитка на тренинг скупу података, док губитак на валидационом скупу података такође опада у првим епохама, али затим почиње да расте након треће епохе. Овај пораст валидационог губитка указује на почетак преобучавања, где модел почиње да претерује са адаптацијом на тренинг податке.

Истовремено, тачност на тренинг сету расте током свих 10 епоха, док је тачност на валидационом сету највиша око четврте епохе, након чега опада. Ово указује да, упркос побољшању на тренинг подацима, модел не успева да одржи исту меру генерализације на валидационим подацима, што даље потврђује присуство преобучавања.

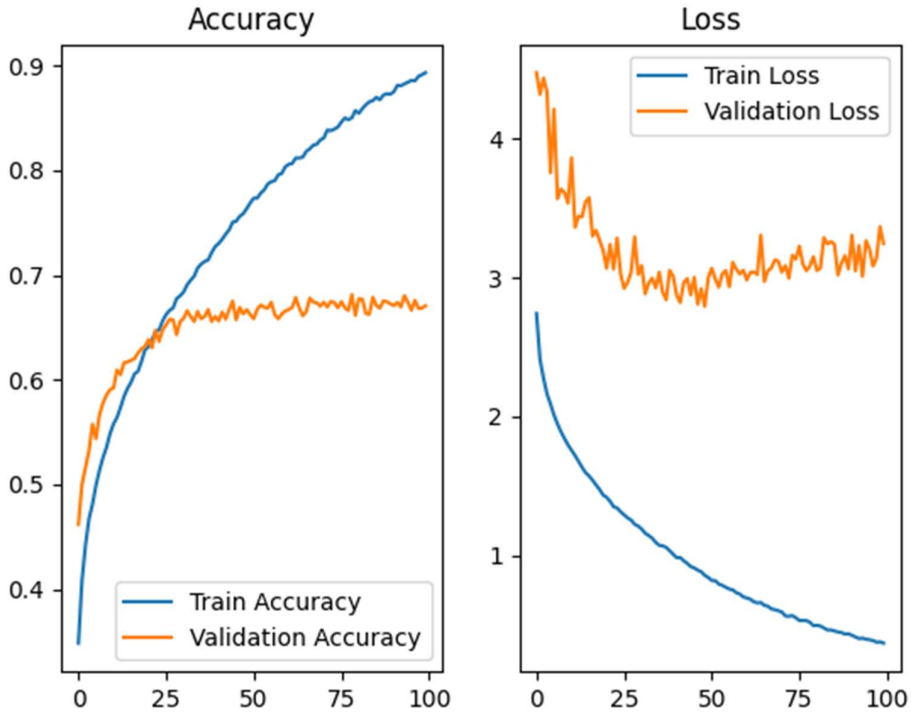


Слика 16 – Први пролаз модела

7.2. Напредак модела у другом пролазу

На другој слици, која приказује резултате тренинга након 100 епоха, може се уочити јасан тренд побољшања како у погледу тачности, тако и у смањењу губитка на тренинг скупу података. Тачност на тренинг подацима досеже скоро 90% након 100 епоха, што показује да модел успешно учи из података током дужег периода тренинга.

Међутим, тачност на валидационом сету стабилизује се око 65% после око 25 епоха и не показује значајан пораст током каснијих епоха. Губитак на валидационом сету такође остаје релативно константан након почетног смањења, што указује да модел не успева да додатно побољша генерализацију, али успева да одржи стабилне резултате током времена без даљег погоршања.



Слика 17 - Други пролаз модела

7.3. Анализа резултата

Модел показује знаке преобучавања након само неколико епоха, што указује на потребу за коришћењем метода за сузбијање преобучавања, као што су рано заустављање, повећање регуларизације (енгл. *regularization*) или примену техника аугментације података у још већој мери. Након почетног пораста, тачност на валидационом сету се стабилизује око 65%, што показује да модел има ограничену способност да се генерализује на нове податке.

Ово може указивати на потребу за даљим прилагођавањем архитектуре модела или оптимизационих хиперпараметара. Упркос овим изазовима, модел показује значајна побољшања у тачности током тренинга, што потврђује ефикасност трансферног учења за класификацију слика хране. Модел успешно користи претходно научене карактеристике из *InceptionV3* модела за адаптацију на специфичан скуп података, што је посебно важно у контексту рада са сликама које садрже "бучне" (неисправне) ознаке.

На основу анализе резултата, можемо закључити да предложени модел показује добру способност учења из података уз помоћ трансферног учења, али такође се јавља и изазов преобучавања и ограничена генерализација на валидационом сету. Додатни рад на оптимизацији хиперпараметара и борби против преобучавања би могли да побољшају укупне резултате модела и омогуће још бољу класификацију слика хране.

8. Закључак

Циљ овог рада био је да се истраже и прикажу могућности примене конволуционих неуронских мрежа (*CNN*) у класификацији слика, са посебним освртом на коришћење техника дубоког учења. Током истраживања, детаљно је објашњена архитектура *CNN*-а, као и његова примена у различитим областима у којим би овакве технике биле од изузетног значаја. Кроз имплементацију модела базираног на *InceptionV3*, рад је показао како трансферно учење може значајно убрзати процес обуке и побољшати прецизност модела.

Упркос оствареним резултатима, уочени су изазови, посебно у погледу преобучавања модела, што указује на потребу за даљим усавршавањем архитектуре и хиперпараметара. Могуће је даље истраживање техника регуларизације и аугментације података како би се побољшала генерализација модела на непознатим подацима. Овај рад доприноси бољем разумевању употребе *CNN*-а у класификацији слика, отварајући могућности за даљи развој у примени ових технологија у различитим индустријама.

Овај рад показује да су савремени алати и технике дубоког учења изузетно ефикасни у задацима класификације слика, као што је препознавање различитих врста хране. Постављени модел постиже високе перформансе и може се применити у бројним апликацијама у индустрији, укључујући системе за препознавање и класификацију хране у реалном времену, апликације за праћење исхране, као и аутоматизоване системе за управљање менијима и састојцима.

Током целог рада је приказано коришћење конволуционих неуронских мрежа и техника трансферног учења за класификацију слика у специфичном домену препознавања хране. Кроз коришћење претходно обученог модела *InceptionV3* и прилагођавање новим подацима, систем је показао солидне резултате, са тачношћу од 90% на тренинг скупу података и 65% на валидационом скупу података. Ипак, модел је показао знакове преобучавања, што указује на потребу за применом додатних техника регуларизације и аугментације података.

Успешно имплементиран систем омогућава даљу примену у задацима класификације слика и може бити проширен на друге домене, као што су сегментација слика или детекција објеката. Потенцијал за проширење постоји у коришћењу различитих архитектура *CNN*-а или додавању података који укључују више класа и варијација слика.

Систем развијен у овом раду може бити даље усавршаван кроз оптимизацију хиперпараметара, додавање нових скупова података и фино подешавање модела. Поред тога, могућности интеграције са различитим апликацијама отварају простор за комерцијалну примену, посебно у индустријама које се ослањају на анализу слика и рачунарски вид.

У будућим истраживањима, постоји потенцијал за даље усавршавање модела применом сложенијих архитектура, техника аугментације података и преносног учења, као и испитивање модела у условима стварне примене. Ова истраживања могу додатно унапредити тачност и робусност модела у разним сценаријима примене.

Дубоко учење представља изузетно моћан алат, успешна примена захтева непрекидно усавршавање и прилагођавање модела специфичностима проблема. Уз адекватну обуку и оптимизацију, ови модели могу значајно унапредити многе индустрије, али је истовремено важно препознати њихова ограничења и критички приступити њиховој примени. Дубоко учење није коначна технологија, већ корак ка будућности у којој ће машине све више постајати помоћници човечанству, али са тим долази и одговорност да се те технологије развијају на начин који је поуздан и етичан.

Литература

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). *TensorFlow: A system for large-scale machine learning*.

[2] Chollet, F. (2015). *Keras* https://keras.io

[3] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition*.

[4] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *Imagenet classification with deep convolutional neural networks*.

[5] Shorten, C., & Khoshgoftaar, T. M. (2019). *A survey on image data augmentation for deep learning*.

[6] Ioffe, S., & Szegedy, C. (2015). *Batch normalization: Accelerating deep network training by reducing internal covariate shift*.

[7] Bossard, L., Guillaumin, M., & Van Gool, L. (2014). *Food-101–mining discriminative components with random forests*.

[8] Binary image classfier: Binary Image classifier CNN using TensorFlow | by Sai Balaji | Techiepedia | Medium, приступано у августу 2024.

[9] Convolutional Neural Network: Convolutional Neural Network - Arun Kumar (arunkrweb.github.io) , приступано у августу 2024.

[10] Документација за *TensorFlow*: All symbols in TensorFlow 2 | TensorFlow v2.16.1, приступано у августу 2024.

Списак скраћеница

ARP	*Application Programming Interface*
CNN	*Convolutional Neural Network*
GPU	*Graphics Processing Unit*
ReLU	*Rectified Linear Unit*
RGB	*Red Green Blue*
RMSP	*Root Mean Square Propagation*
SGD	*Stochastic Gradient Descent*
VGG	*Visual Geometry Group*

Списак слика