



Roope Jokelainen

# Tekoälyn hyödyntäminen videopelin kehityksessä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

16.5.2023

# Tiivistelmä

Tekijä: Roope Jokelainen  
Otsikko: Tekoälyn hyödyntäminen videopelin kehityksessä  
Sivumäärä: 32 sivua  
Aika: 16.5.2023

Tutkinto: Insinööri (AMK)  
Tutkinto-ohjelma: Tieto- ja viestintätekniikka  
Ammatillinen pääaine: Pelisovellukset  
Ohjaaja: Lehtori Miikka Mäki-Uuro

---

Insinööriyössä selvitettiin mahdollisuutta hyödyntää tekoälyä videopelin kehityksessä. Insinööriyössä kehitettiin Othello-lautapelistä videopeli käyttäen Unity-pelimoottoria. Kehityksessä tekoäly teki videopelin kehityssuunnitelman, jonka mukaan videopeli kehitettiin. Tekoäly lisäksi tuotti kehityksessä käytetystä koodista suurimman osan.

Tekoäly on ohjelma tai kone, joka ratkaisee ongelmia. Ongelmat vaativat ihmismäistä älyä suorituksessa. Tekoälyt luokitellaan vahvoihin tekoälyihin ja heikkoihin tekoälyihin. Vahvat tekoälyt osaavat ratkaista minkä tahansa ongelman ilman koulutusta ongelmaan. Heikot tekoälyt osaavat vain ratkaista tietyn ongelman.

Tekoälyjä on kehitetty ratkaisemaan ongelmia, joiden ratkaiseminen on ihmisille vaikeaa. Niitä on kehitetty algoritmeja käyttäen tai koneoppimista hyödyntäen. Algoritmit ovat perinteinen tapa kehittää tekoälyä, jossa ohjeistetaan ongelman ratkaisuun. Koneoppimisessa tekoäly kehitetään erilaisilla menetelmillä, joissa yhteistä on tietoaistoin antaminen erilaisille oppimismalleille, joista mallit osaavat tehdä johtopäätöksiä.

Insinööriyössä videopeliin kehitettiin tekoäly Minimax-algoritmia hyödyntäen, jossa luodaan pelitilanteita, jotka pisteytetään erilaisten muuttujien avulla. Muuttujat riippuvat pelistä ja tekoälyn tehokkuuden säätämisestä. Othello-videopelissä tekoälyn muuttujat olivat pelinappuloiden määrä pelilaudalla, nappulan saanti kulmaan, nappulan saanti reunaan ja pelin voittaminen. Minimax-algoritmi on tekoäly, jonka tehokkuus perustuu muuttujiin mutta lisäksi siihen, kuinka syvälle pelitilanteita käytiin pisteyttämässä.

ChatGPT-tekoälyä hyödynnettiin pelinkehityksessä. Se osaa kirjoittaa kaikenlaista tekstiä monilla eri kielillä, mutta lisäksi se osaa kirjoittaa monilla eri ohjelmointikielillä toimivaa koodia. ChatGPT:n ohjelmointiosaamista hyödynnettiin videopelin kehityksessä. ChatGPT:n kirjoitukset eivät ole täydellisiä, vaikka ovatkin ihmismäisiä, jolloin varsinkin ohjelmoinnissa virheiden löytäminen on vaikeaa.

Avainsanat: tekoäly, koneoppiminen, ChatGPT, Minimax

---

Tämän opinnäytetyön alkuperä on tarkastettu Turnitin Originality Check -ohjelmalla.

## Abstract

Author: Roope Jokelainen  
Title: Utilisation of artificial intelligence in game development  
Number of Pages: 32 pages  
Date: 16 May 2023

Degree: Bachelor of Engineering  
Degree Programme: Information and Communications Technology  
Professional Major: Game Applications  
Supervisor: Miikka Mäki-Uuro, Senior Lecturer

---

The purpose of this final year project was to find is it possible to utilise artificial intelligence in game development. The final year project was conducted by developing Othello board game in unity game engine utilising ChatGPT. ChatGPT is artificial intelligence developed by OpenAI that can write many different types of texts or even write usable code.

ChatGPT is developed using machine learning. Machine learning works as giving large dataset to machine go through and find conclusions about data using different methods. Method used in ChatGPT is called transformer in which data is transferred to numbers that it reorganizes order of numbers based of what it needs to answer.

ChatGPT in final year project did most of the programming and development of artificial intelligence to game utilising Minmax algorithm. Minimax algorithm work is based to giving points to every move in game and depending on whose move it is most point given move or least is returned to computer and that move is executed. The final year project proves that artificial intelligence can be used in game development. In conclusion artificial intelligence can speed up development but also less people needed for development.

Keywords: Artificial intelligence, Machine Learning, ChatGPT, Minimax

# Sisällys

1	Johdanto	1
2	Tekoäly	2
2.1	Klassinen tekoäly	2
2.2	Koneoppiminen	4
2.2.1	Neuroverkot	6
2.2.2	Syväoppiminen	8
3	ChatGPT-kielimalli	11
4	Videopelin kehittäminen tekoälyn avulla	17
4.1	Othello-peli	17
4.2	Kehitystyö	19
4.3	Yhteenveto ChatGPT-tuotannosta	25
5	Johtopäätökset	26
	Lähteet	29

## 1 Johdanto

Insinöörityössä selvitetään, pystyykö tekoälyä hyödyntämään pelinkehityksessä. Työssä kehitetään Unity-pelimoottorissa Othello-peli, johon luodaan tekoäly tekoälyllä. Tekoälyssä hyödynnetään Minimax-algoritmia. Pelinkehityksessä hyödynnetään ChatGPT-tekoälyä, joka osaa kirjoittaa usealla eri kielellä esseitä, runoja, lakitekstejä ja kirjeitä tai ohjelmoida usealla eri ohjelmointikielellä.

Insinöörityössä tutkitaan tekoälyä, mitä se on ja miten sitä käytetään tietotekniikassa. Työssä tutkitaan erityisesti tekoälyn kehittämistä, kuten koneoppimista ja sen osa-alueita, kuten neuroverkkoja ja syväoppimista. Kun tekoälyyn on perehdytty, tutkitaan ChatGPT:n toimintaa ja sitä, miten se on kehitetty. Insinöörityö on poikkeuksellinen, koska siinä hyödynnetään tekoälyä, joka pystyy suorittamaan suuren osan työstä, mutta lisäksi pelisovellusten pääaineen koulutusta. Tekoälyn käyttö insinöörityössä on kyseenalaista, mutta koska ChatGPT on ajankohtainen, oli mahdollista työstää insinöörityö ChatGPT:stä.

Insinöörityön tavoitteena on selvittää, onnistuuko Othello-pelin kehittäminen käyttämällä ChatGPT-tekoälyä. Sen lisäksi tavoitteena on selvittää mahdollisia ongelmia, joita tekoälyjen käyttö aiheuttaa pelinkehityksessä ja miten mahdollisia ongelmia voidaan ehkäistä. Insinöörityössä tutkitaan tekoälyn vaikutuksia työllisyyteen sekä vaatimuksia tekoälyn käytössä pelinkehityksessä ja muussa ohjelmointikehityksessä. Lisäksi insinöörityössä tutkitaan tekoälyjen kehittymistä ja niihin liittyvää sääntelyä tai lainsäädäntöä sekä hyviä tapoja luoda sääntöjä, jotka toimivat kaikkien tekoälyjen kanssa. Insinöörityössä tarkastellaan lisäksi tekoälyn mahdollisuuksia tulevaisuudessa ja sen roolia yhteiskunnassa, jossa tekoälyt ovat yleisessä käytössä.

## 2 Tekoäly

### 2.1 Klassinen tekoäly

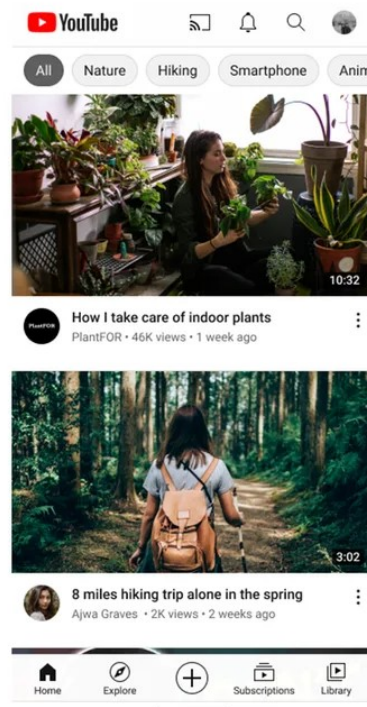
Tekoäly tarkoittaa laitetta tai ohjelmaa, jonka tarkoituksena on ratkaista ongelmia, jotka vaativat ihmismäistä älyä suoriutumisesta (1; 2). Tekoälyt luokitellaan kahteen luokkaan: vahvoihin tekoälyihin ja heikkoihin tekoälyihin.

Vahvat tekoälyt ovat tekoälyjä, jotka pystyvät ratkaisemaan ongelmia, joihin niitä ei ole opetettu. Tällaisia tekoälyjä nähdään ainoastaan elokuvissa tai muussa viihteessä, sillä tällaisten tekoälyjen kehittäminen ei ole vielä onnistunut. (1; 2.)

Heikot tekoälyt ovat vastaavasti tekoälyjä, jotka osaavat ratkaista vain opetetun ongelman. Esimerkiksi IBM:n kehittämä Deep Blue -shakkitietokone, joka voitti vuonna 1997 shakin maailmanmestarin Garri Kasparovin ja oli ensimmäinen tietokone, joka voitti shakin maailmanmestarin normaaleilla turnauspeliajoilla, on tällainen tekoäly. (3.)

Tekoälyt ovat nykyelämässä arkipäivää, ja ihmiset käyttävät niitä huomaamattaan. Esimerkiksi navigaattori antaa reitin, joka on luotu tekoälyn avulla, ja YouTube videoiden suositukset ovat tekoälyn luomia. Navigaattorin tekoäly on ohjelmoitu niin, että ohjelmoija on kertonut ohjelmalle, minkä mukaan reitit luodaan. YouTube videoiden suositusten tekoäly on jatkuvasti oppiva tekoäly, joka yrittää oppia jokaisen käyttäjän videomieltymykset erilaisten tietojen

perusteella (4). Kuva 1 näyttää YouTube-videoiden suositukset YouTubeissa.



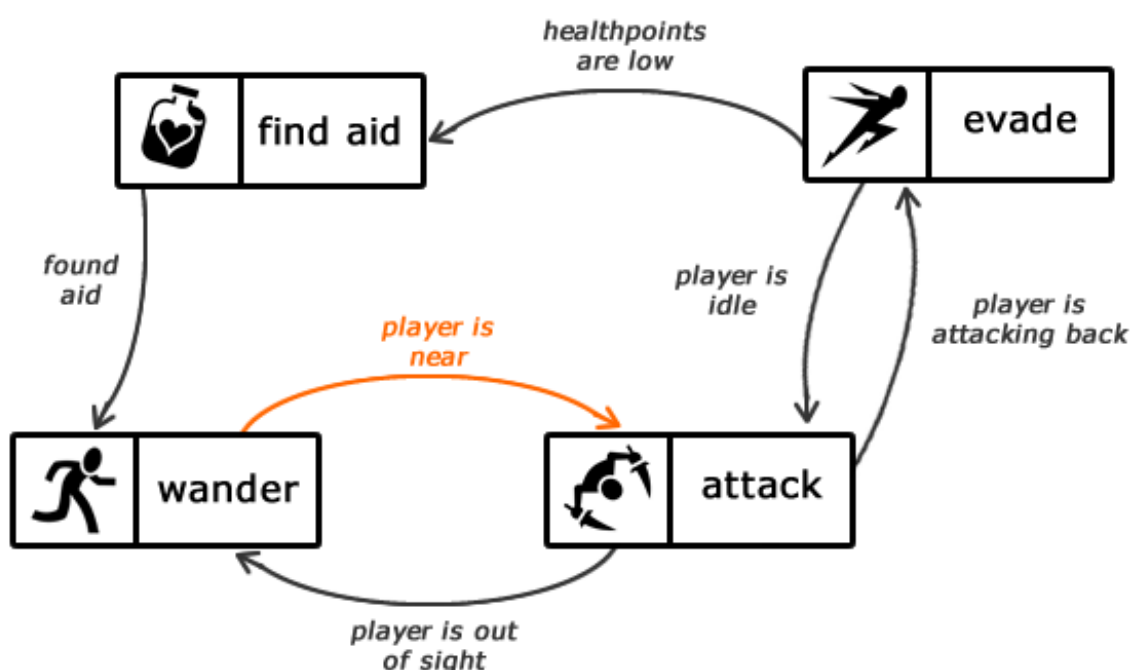
Kuva 1. YouTube-videoiden suositusverkkosivu (4).

Tekoälyjen kehittämisessä pitäisi Isaac Asimovin mukaan ottaa huomioon, että ne eivät saisi kehittyä itsestään, vaan niiden kehitys pohjautuu yhteiseen arvo-maailmaan. Hän loi kolme lakia, jotka kuvaavat tätä arvopohjaa. Ensimmäinen laki määrittää, että tekoäly ei saa vahingoittaa ihmistä tai toimia siten, että se aiheuttaa haittaa ihmiselle. Toiseksi tekoälyn täytyy noudattaa ihmisen ohjeita, paitsi jos ne rikkovat ensimmäistä lakia. Kolmanneksi tekoälyn täytyy suojella itseään rikkomatta ensimmäistä ja toista lakia. (1; 5.)

Tekoälyt ovat kaikkialla yhteiskunnassa, ja ne ohjaavat autoja ja drooneja. Tekoälyt lisäksi kääntävät kieliä, ja niiden avulla etsitään uusia lääkkeitä. Tekoälyjen avulla ennakoidaan kulttuurista kiinnostusta. Tekoälyjä hyödynnetään massadatan käsittelyssä, jonka avulla yritykset voivat parantaa tuotteitaan kerätyn datan perusteella. Tulevaisuudessa valtiot voivat hyödyntää massadataa esimerkiksi ihmisten toiminnan seuraamiseen ja mahdollisen rikollisuuden estämiseen ennen rikoksen tapahtumista. Tällainen yhteiskunta vaatii paljon osaajia,

jotka voivat luoda tarvittavia tekoälyjä ja käsitellä tekoälyjen tuottamia ennusteita tai muita massadatan perusteella tehtyjä päätelmiä. (1.)

Tekoälyä esiintyy runsaasti videopeleissä, sillä peleissä on yleensä vihollisia, jotka ovat tekoälyjä. Vihollisten lisäksi peleissä on lisäksi ystäviä, jotka auttavat pelaajaa läpäisemään pelin. Kauppiaat myyvät varusteita, jotka auttavat pelaajaa pelissä, ja avoimen maailman peleissä on lisäksi sivuhahmoja, joiden merkitys pelissä on luoda tunne oikeasta maailmasta. Usein peleissä esiintyy lisäksi eläimiä, jotka eivät ole vihollisia. Videopelien tekoälyt toteutetaan yleensä tilakoneilla tai algoritmeilla, mutta lisäksi koneoppimista hyödynnetään niiden luomiseen. Kuva 2 havainnollistaa vihollisen tilakonetta videopelissä.



Kuva 2. Esimerkki tilakoneesta videopelissä (6).

## 2.2 Koneoppiminen

Koneoppiminen on tekoälyn osa, joka keskittyy käyttämään tietoa ja algoritmeja imitoimaan ihmisen oppimista, jolloin se vähitellen oppii tunnistamaan kaavoja ja säännönmukaisuuksia datassa ja parantamaan ennusteiden tarkkuutta. Koneoppiminen on tärkeä osa datatiedettä, joka on jatkuvasti kasvava ala.

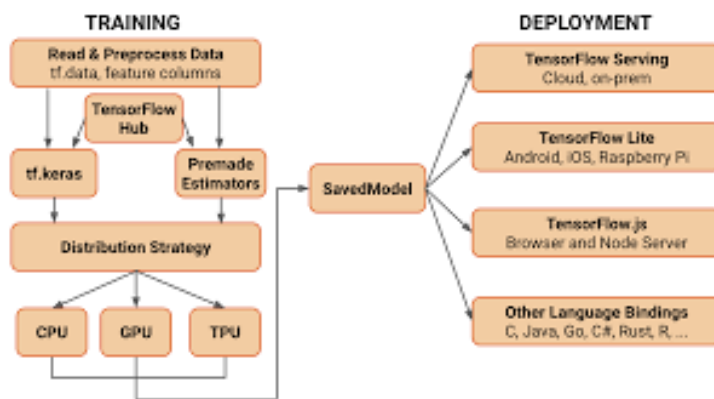


Koneoppimisessa kone oppii käyttämällä tilastoja, jotka algoritmien kouluttamana luokittelevat tai ennustavat haluttuja tietoja, jolloin löydetään tärkeitä tietoja tiedonlouhintaprojekteista (7). Tällaiset tiedot voivat olla ajava voima, jonka perusteella kehitetään sovelluksia tai yrityksiä. Koneoppimisessa tavoitteena on löytämään säännönmukaisuuksia tiedoista mutta lisäksi myös sellaisia yhteyksiä eri muuttujien välillä, joita ihmiset eivät löydä.

Koneoppimisen menetelmiä ovat ohjattu oppiminen, ohjaamaton oppiminen, osittain ohjattu oppiminen ja vahvistusoppiminen. Ohjattu oppiminen perustuu siihen, että tietoaaineisto on nimiöity, ja sen avulla algoritmit luokittelevat tietoa tai ennustavat ulostulojen tarkkuutta. Ohjaamattomassa oppimisessä tietoa ei ole nimiöity eikä siihen muutenkaan kosketa, vaan algoritmit analysoivat ja ryhmittävät tietoaaineistoa. Osittain ohjattu oppimisen menetelmä on ohjatun ja ohjaamattoman väliratkaisu, joka koulutetaan pienillä lokeroituilla tietoaaineistoilla, minkä jälkeen se osaa lisäksi käsitellä lokeroimatonta tietoaaineistoa. (7.) Vahvistusoppiminen on samanlainen kuin ohjattu oppiminen, mutta sitä ei ole koulutettu malliaineistolla vaan yrityksen ja erehdyksen kautta, jossa onnistumisista palkitaan ja epäonnistumisista rangaistaan. Koneoppimisalgoritmeja luodaan integroidun ohjelmoinnin tukiympäristöjä käyttäen, joita ovat TensorFlow (8) ja PyTorch (9).

TensorFlow on avoimen lähdekoodin ohjelmakirjasto, jota käytetään numeeriseen laskentaan ja laajan skaalan koneoppimiseen. TensorFlow'n on kehittänyt Google Brain -työryhmä. TensorFlow käyttää Python- tai JavaScript-ohjelmointikieliä luodakseen kätevän julkisivun API-rajapinnan sovellusten luomiseen. Se suorittaa ohjelmia tehokkaalla C++-ohjelmointikielellä. (10.) Kuva 3

havainnollistaa TensorFlow-toiminnan logiikkaa.



Kuva 3. TensorFlow 2.0:n toimintakaavio (11).

PyTorch on avoimen lähdekoodin ohjelmakirjasto, joka on kehitetty tarjoamaan laajempaa joustavuutta ja nopeamman syväoppimisverkkojen luomisen kuin edeltäjänsä Torch. PyTorchin on kehittänyt Meta, joka tunnetaan Facebookista. PyTorch käyttää Pythonia, ja on mahdollista suorittaa vain osia koodista reaaliajassa. (12.) Kuva 4 havainnollistaa PyTorchin työkulkua koneoppimiseen.

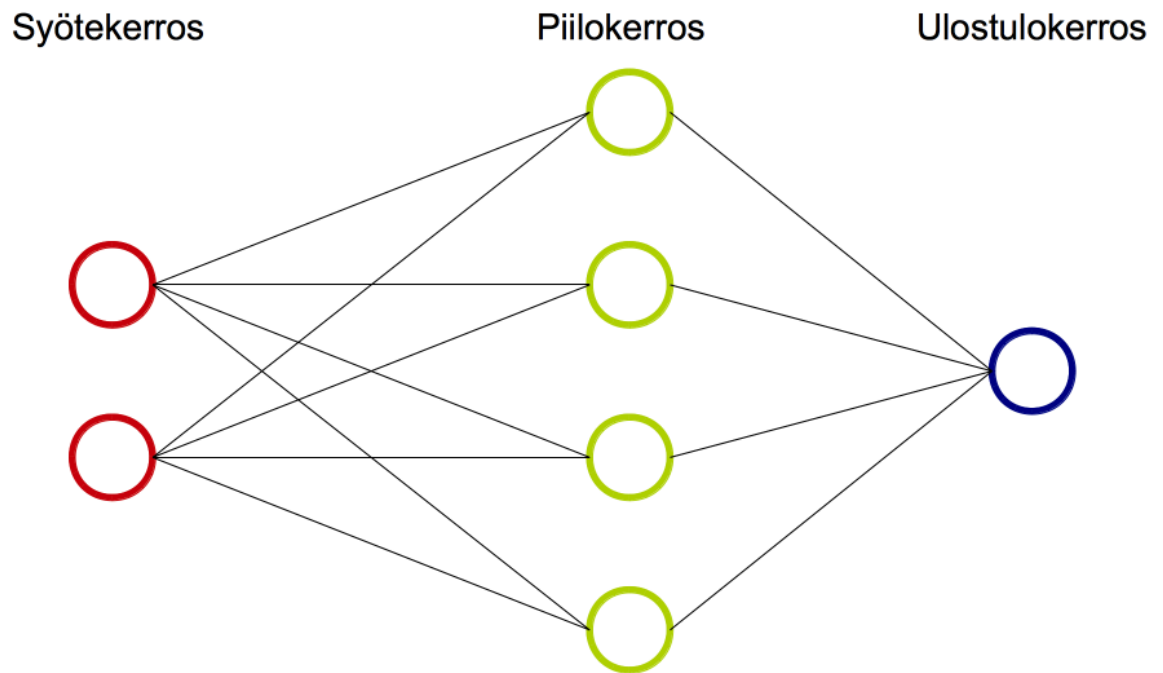


Kuva 4. PyTorchin tekoälyn kehitysvaiheet (13).

### 2.2.1 Neuroverkot

Neuroverkko on koneoppimisen osa-alue, jolla tarkoitetaan menetelmää, jolla opetetaan tietokonetta prosessoimaan tietoa mallintamalla ihmisen aivoja (14). Neuroverkot koostuvat solmuista tai hermosoluista, jotka yhdistyvät toisiinsa

muistuttaen aivoja. Tällöin luodaan sopeutuva järjestelmä, jossa kone oppii virheistään ja kehittyy jatkuvasti. Neuroverkoissa jokaisella solulla on syötetieto, painokerroin, kynnyks ja ulostulo. (15.) Neuroverkot yrittävät ratkaista monimutkaisia ongelmia, kuten dokumentaation tiivistäminen tai kasvojen tunnistaminen. Kuva 5 esittää yksinkertaisen neuroverkon, jossa on syötekerros, piilokerros ja ulostulokerros.



Kuva 5. Yksinkertainen neuroverkon kuvaaja (16).

Neuroverkkoja tehdään silloin, kun tarvitaan ohjelmaa, joka pystyy tekemään järkeviä päätöksiä vähällä ihmisen avustuksella, sillä neuroverkot oppivat ja mallintavat yhteyksiä sisään- ja ulostulotiedon välillä, jotka ovat usein epälineaarisia ja monimutkaisia (14).

Neuroverkkoja käytetään silloin, kun tarvitaan tietokoneelle tietoa ja käsitystä kuvista ja videoista. Esimerkiksi itseajava auto pystyy havainnoimaan liikenne-merkkejä ja muita liikenteen käyttäjiä. Neuroverkkoja käytetään lisäksi silloin, kun tarvitaan puheen tunnistusta, kuten esimerkiksi Microsoftin Cortana, joka tunnistaa puhetta ja suorittaa käyttäjän tehtäviä. Luonnollisen kielen

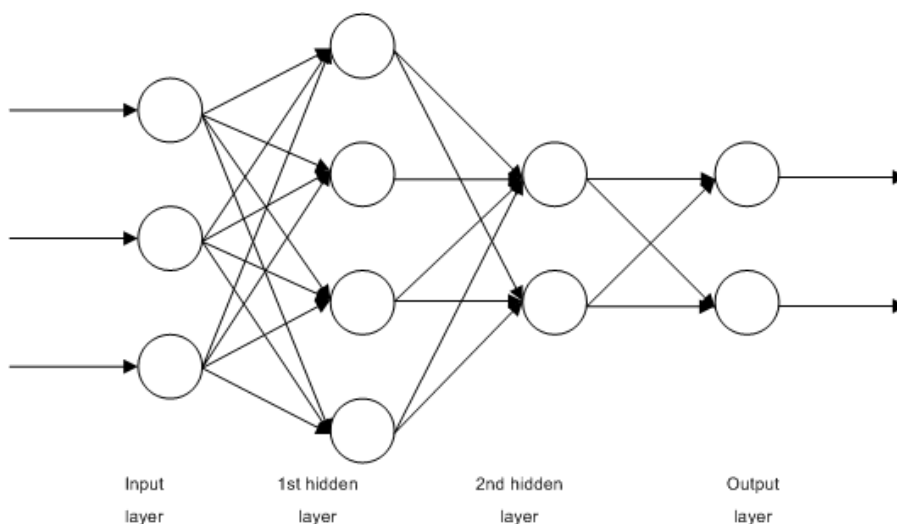
käsittelyjärjestelmässä lisäksi hyödynnetään neuroverkkoja, kun tarvitaan ymmärrystä kirjoituksesta tai asiakirjasta, kuten erilaisissa virtuaalisissa asiakaspalvelijoissa.

Neuroverkot rakentuvat kerroksista, jotka ovat syötekerros, piilokerros ja tulo-kerros. Syötekerros saa ulkomaailmasta tietoa, jonka se käsittelee, analysoi tai luokittelee ja jakaa sen seuraavalle kerrokselle. Piilokerros saa syötekerroksen tai toisen piilokerroksen tiedot, joita se käsittelee ja jakaa eteenpäin. Tulokerros antaa viimeisen tuloksen neuroverkkojen käsittelystä, ja se voi koostua yhdestä tai useammasta solusta riippuen siitä, onko kyse binäärisestä luokittelusta vai moniluokkaisesta luokittelusta. Binäärisessä luokittelussa on vain yksi solu tulokerroksessa, kun taas moniluokkaisessa luokittelussa voi olla useampia soluja kuin yksi. (14.)

### 2.2.2 Syväoppiminen

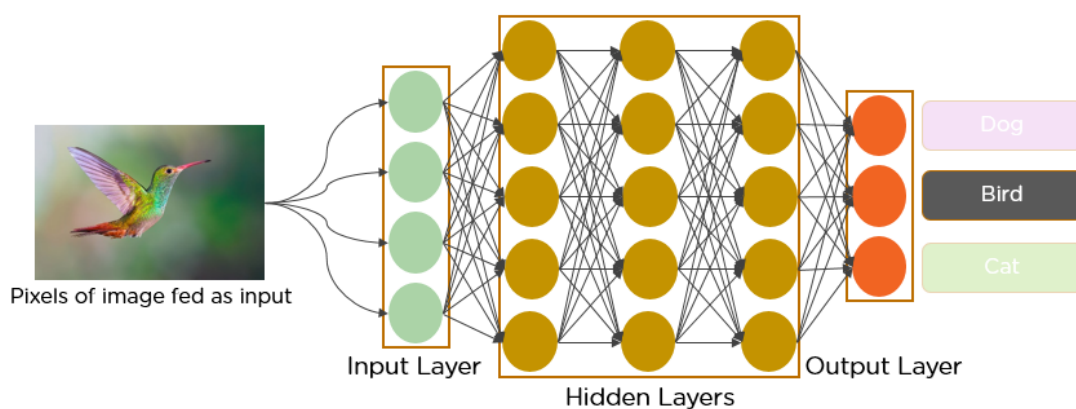
Syväoppiminen on neuroverkkojen osa-alue. Jos neuroverkoissa on vähintään kolme kerrosta, puhutaan syväoppimisesta. Syväoppimista käytetään, kun käsitellään isoja aineistoja ja halutaan tarkkoja ja optimoituja ennustuksia, joissa syötteen ja ennustettavan muuttujan välinen yhteys on monimutkainen. (17.)

Kuva 6 esittää syväoppimisen neuroverkkoa, jossa on kaksi piilokerrosta.



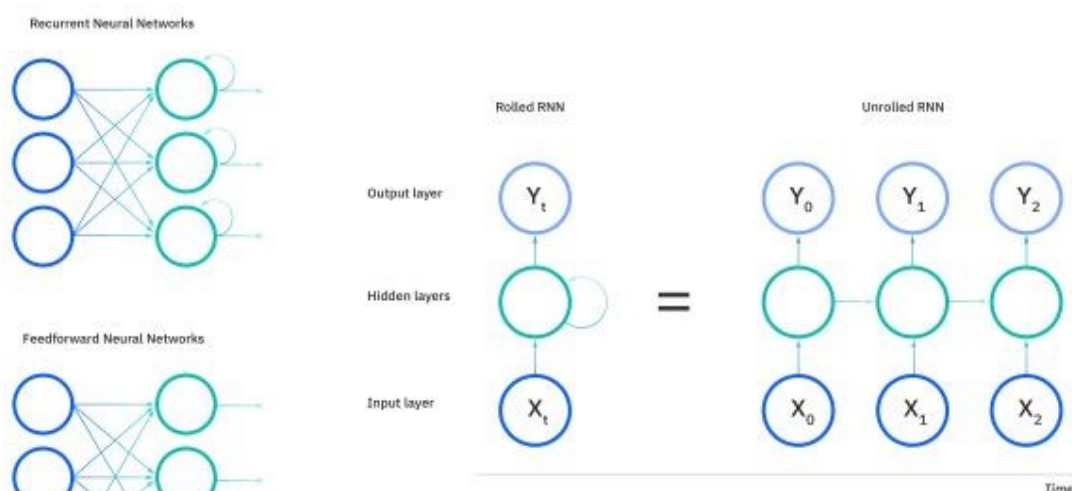
Kuva 6. Syväoppivan neuroverkon kuvaaja (18).

Konvoluutioneuroverkkoa käytetään esimerkiksi esineiden havaitsemiseen ja tunnistamiseen. Konvoluutioisessa neuroverkossa on tavallisten neuroverkkojen kerrosten lisäksi kolme pääkerrosta: konvoluutioinen kerros, koontikerros ja täysin yhdistetty kerros. Konvoluutioinen kerros vaatii syöttötiedoksi yleensä kuvan tai videon, mutta lisäksi suodattimen ja piirrekartan. (19.) Koontikerros vähentää monia parametreja syöttötiedosta, mikä auttaa poistamaan monimutkaisuutta, parantamaan tehokkuutta ja vähentämään mahdollista ylisovittamista eli mallin sopeutumista liian hyvin tietoon, mikä ei ole haluttu tulos (20). Täysin yhdistetyssä kerroksessa jokainen tuloskerroksen solu yhdistyy aikaisemman kerroksen soluun. Kuva 7 havainnollistaa konvoluuttista neuroverkkoa.



Kuva 7. Konvoluuttisen neuroverkon kuvaaja (21).

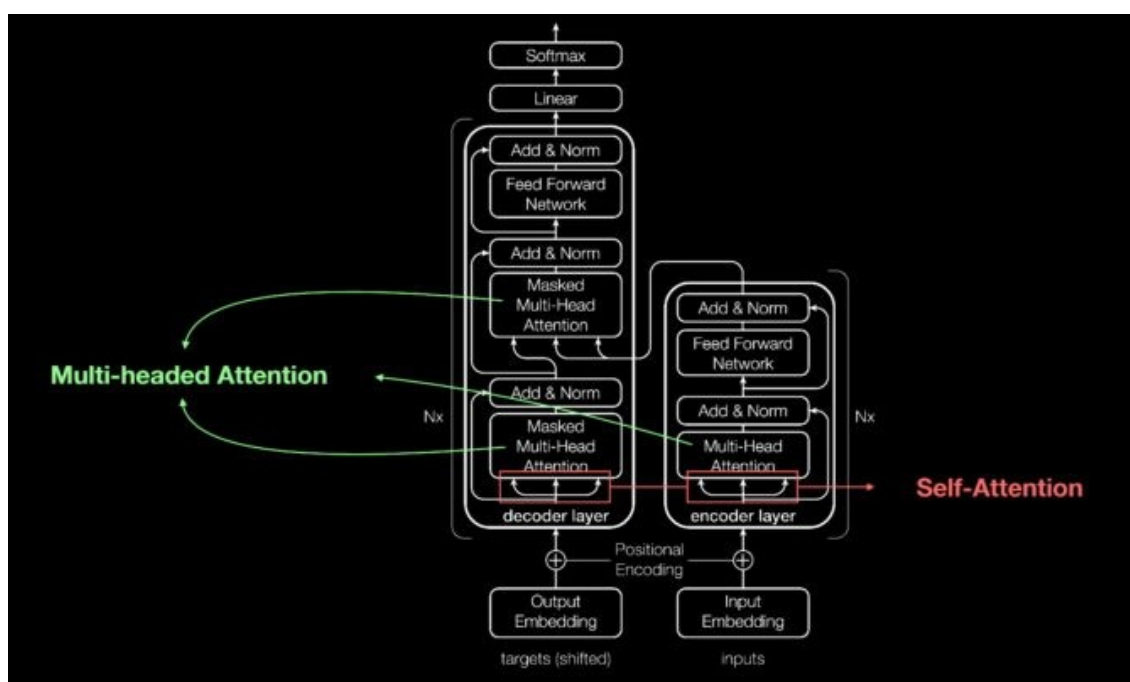
Takaisinkytkentäinen neuroverkko käyttää peräkkäistä tietoa tai aikasarjan tietoa, mikä on hyödyllistä esimerkiksi kielten kääntämisessä. Takaisinkytkentäinen neuroverkko pystyy ottamaan huomioon aiemmat syötteet, jotka vaikuttavat sen nykyiseen syötteeseen ja tulosteeseen, mikä tekee siitä riippuvaisen aikaisemmista aikasarjoista. Takaisinkytkentäinen neuroverkko koostuu vain tavallisista neuroverkon kerroksista, joihin sillä on jatkuva pääsy. (22; 23.) Kuva 8 havainnollistaa takaisinkytkettyvän neuroverkon toimintaa.



Kuva 8. Takaisinkytkettyyn neuroverkon kuvaaja (22).

Muuntaja eli englanniksi transformer on neuroverkko, joka on melkein siviilitantanut konvoluutioisen neuroverkon ja takaisinkytkettyyn neuroverkon (24; 25). Tämä johtuu muuntajan kyvystä käsitellä lähes mitä tahansa tietoa ja suuria määriä, mikä ei ole mahdollista konvoluutioisessa neuroverkossa ja takaisinkytkettyssä neuroverkossa, koska ne vaativat tiedon nimeämistä. Muuntajan tehokkuus perustuu kooderiin, joka sijoittaa esimerkiksi sanat oikeaan järjestykseen. OpenAI:n kehittämä ChatGPT pohjautuu muuntajaan. Kuva 9

havainnollistaa muuntajan toimintaa tilakoneena.



Kuva 9. Muuntajan toiminnan kuvaaja (25).

### 3 ChatGPT-kielimalli

Insinööriyössä hyödynnettiin kielimallia nimeltä ChatGPT (26), joka osaa kirjoittaa vähintään 95:tä eri kieltä ja lisäksi tuntee monia ohjelmointikieliä ja pelinkkehitysympäristöjä, kuten Unity-pelimoottorin (27). ChatGPT kertoo itsestään seuraavanlaisesti.

"ChatGPT on yksi nykypäivän merkittävimmistä kielimalleista, joka on kehitetty OpenAI:n GPT-3.5-arkkitehtuurilla. Se on luotu oppimaan ja tuottamaan ihmisten kanssa kommunikoivaa luonnollista kieltä, mikä tekee siitä erittäin arvokkaan työkalun monilla aloilla.

ChatGPT on osoittanut olevansa hyödyllinen monilla eri alueilla, mukaan lukien asiakaspalvelu, chatbotit ja käännössovellukset. Se on myös auttanut ihmisiä yhdistämään toisiinsa eri puolilla maailmaa ja edistänyt kulttuurien välistä ymmärrystä.

ChatGPT:n avulla voidaan myös luoda parempia oppimis- ja koulutusmahdollisuuksia. Se voi auttaa opiskelijoita oppimaan uusia kieliä ja parantamaan kielitaitoaan, mikä on erittäin tärkeää yhä globaalimmassa maailmassa. ChatGPT voi myös auttaa opettajia ja

kouluttajia saavuttamaan suuremman yleisön ja tarjoamaan paremman koulutuksen.

Vaikka ChatGPT onkin erittäin hyödyllinen työkalu, on myös tärkeää ottaa huomioon sen rajoitukset. ChatGPT ei ole täydellinen ja sen vastaukset voivat olla epätarkkoja tai epäselviä. On myös tärkeää varmistaa, että sen käyttö on vastuullista ja että sen vastaukset eivät aiheuta haittaa tai vahingoita ketään.

Lisäksi on tärkeää ottaa huomioon, että ChatGPT on vain työkalu, joka tarvitsee ihmisiä sen käyttämiseen. Se ei voi korvata inhimillistä vuorovaikutusta tai ajattelua, ja sen on tarkoitus toimia avustavana työkaluna ihmisten kanssa kommunikoinnissa.

Kaiken kaikkiaan ChatGPT on erittäin hyödyllinen ja monipuolinen työkalu, joka voi auttaa ihmisiä monilla eri aloilla. On tärkeää käyttää sitä vastuullisesti ja ottaa huomioon sen rajoitukset, jotta voimme saavuttaa sen parhaan mahdollisen hyödyn.”

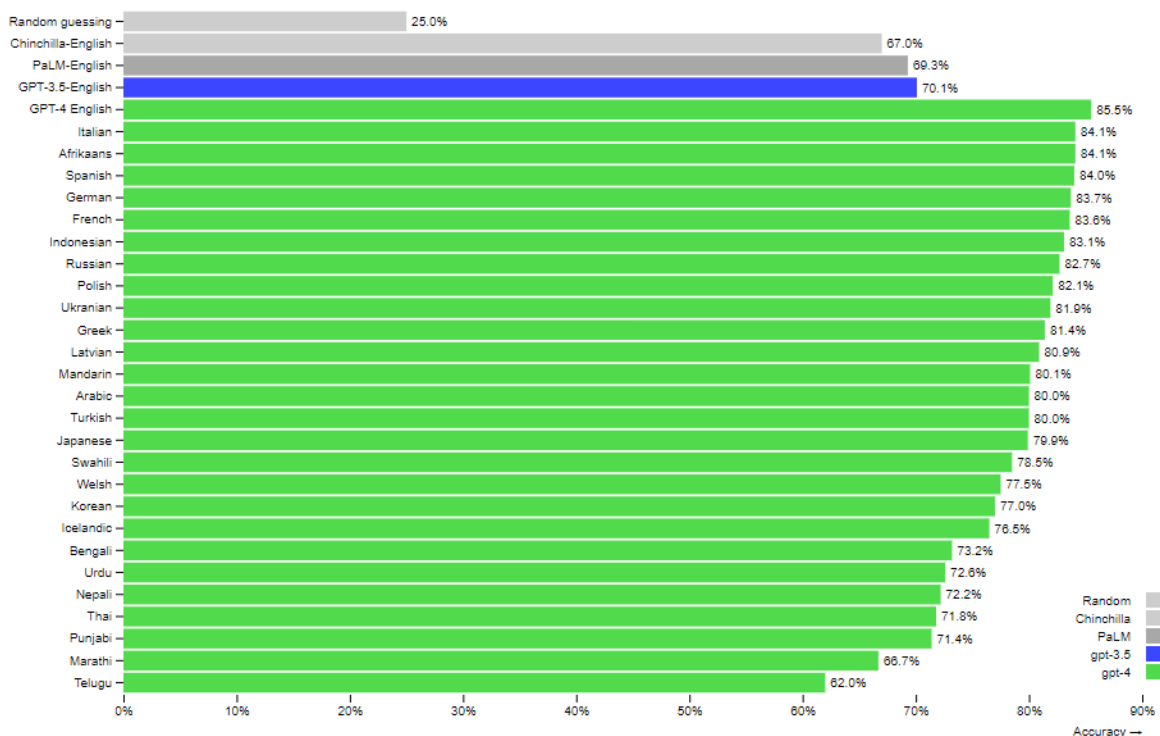
Koska ChatGPT voi olla epätarkka, tarkistetaan sen kirjoitukset. Ensimmäisen kappaleen voidaan katsoa pitävänsä paikkaansa vaikkakin merkittävyyttä voidaan kyseenalaistaa. Toisessa kappaleessa voidaan kyseenalaistaa ihmisten yhdistäminen ympäri maailman, mutta väite on vaikea todistaa vääräksi tai oikeaksi. Kolmannessa kappaleessa kyseenalaista että, opettajat ja kouluttajat saavuttaisivat suuremman yleisön ja tarjoaisivat paremman opetuksen ChatGPT:n avulla, mikä voi pitää paikkansa mutta on vaikea todistaa. Neljäs kappale vaikuttaa pitävän paikkaansa, sillä siinä myönnetään virheellisyys ja mahdollisuus väärinkäyttöön, joka voi vahingoittaa muita. Viides ja kuudes kappale ovat järkevän oloisia, eivätkä ne ole kyseenalaisia.

ChatGPT:stä on lisäksi laajaan tekstuaaliseen opetusaineistoon perustuva neuroverkko. ChatGPT on 3.5-versio GPT-kielimallista, joka on generatiivinen esikoulutettu muuntaja (28; 29). GPT 3 -versio koulutusmateriaalin koko on 45 teratavua, josta koko Wikipedia muodostaa vain 3 %. Koulutuksessa käytettiin 175 miljardia parametriä, jotka ovat painokertoimia, jolla tarkoitetaan havaintoon liittyvä painoa, jossa havainto saadaan vastaamaan perusjoukkoa. Versiossa 4 uskotaan olevan 3–4 kertaa enemmän painokertoimia kuin 3.5-versiossa. ChatGPT:ssä on käytetty ohjattua oppimista ihmisen palautteella, mikä perustuu siihen, että ihmiset pisteyttävät kielimallin luomat tekstit, jotka annetaan takaisin



koneelle. Tällä pyritään estämään rasistinen ulostulo. ChatGPT on noin 70 % oikeassa englanniksi ja GPT 4 on noin 85 % oikeassa. (30.) Kuva 10 esittää ChatGPT:n ja GPT-4:n kielitaitoa eri kielillä.

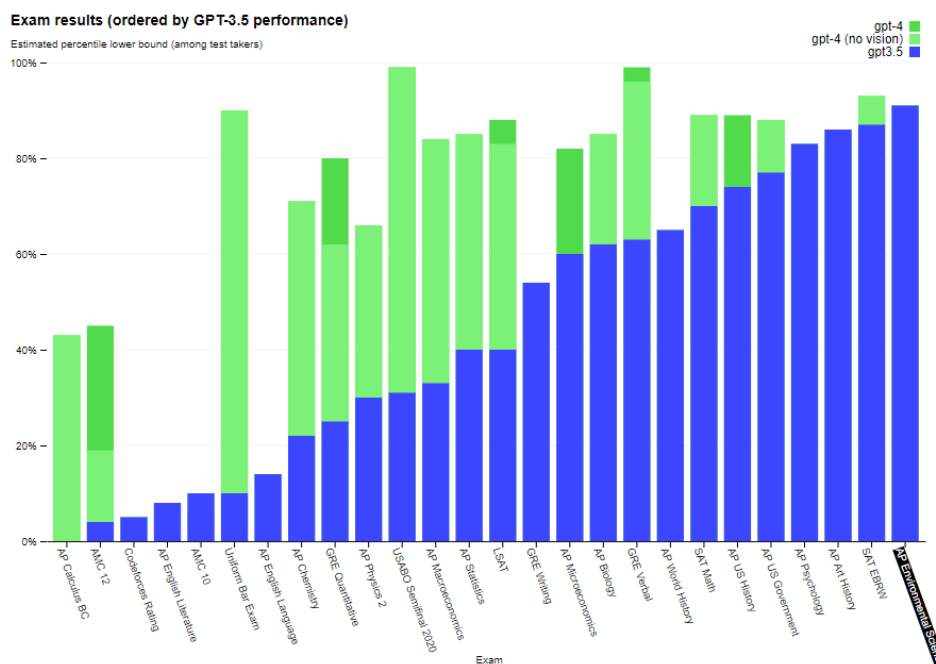
**GPT-4 3-shot accuracy on MMLU across languages**



Kuva 10. ChatGPT:n ja GPT-4:n kielitaito (30).

ChatGPT suoriutuu erilaisista akateemisista ja ammattitaitoa mittaavista kokeista keskimäärin parhaimman 50 %:n joukkoon. GPT 4 taas oli melkein kaikissa kokeissa parhaimman 50 %:n joukossa ja sai muutamia täysiä pisteitä kokeista, joita ChatGPT ei onnistunut saavuttamaan. (30.) Kuva 11 havainnollistaa

## ChatGPT:n ja GPT-4:n suoriutumista kokeista.



Kuva 11. ChatGPT:n ja GPT-4:n suoriutuminen erilaisista kokeista (30).

GPT 4:n onnistuminen perustuu esikoulutusprosessiin eikä ohjattuun oppimiseen ihmisen palautteesta, mistä voidaan päätellä, että sen 3–4 kertaa enemmän olevat parametrit tekevät eron verrattaessa sitä aiempiin versioihin (30).

ChatGPT luo tekstiä perustuen annettuun aiheeseen. Kun se luo tekstiä, se pääättelee sen perusteella, mitä kirjoittaa. Tämän vuoksi se osaa luoda ihmisistä tekstiä, kuten on osoitettu Turingin kokeen läpäisyllä, mikä tarkoittaa kykyä saada ihminen luulemaan keskustelelevansa ihmisen kanssa, vaikka keskustelukumppani olisi itse asiassa kone (31).

ChatGPT:n käyttäminen ei ole ilmaista, koska sähkö ei ole ilmaista, mutta lisäksi se suoritetaan Microsoftin Azure-pilvipalvelussa, joka maksaa noin yhden sentin 30 sanan vastauksesta. Tästä voidaan arvioida, että ChatGPT maksaa noin 100 000 dollaria päivässä, mikä tekee noin 3 miljoona dollaria kuukaudessa ja vuodessa 36 miljoona dollaria. (32.) Tästä syystä OpenAI tarjoaa maksullista palvelua, jossa pääsee aina käyttämään ChatGPT:tä, koska ilmaiset

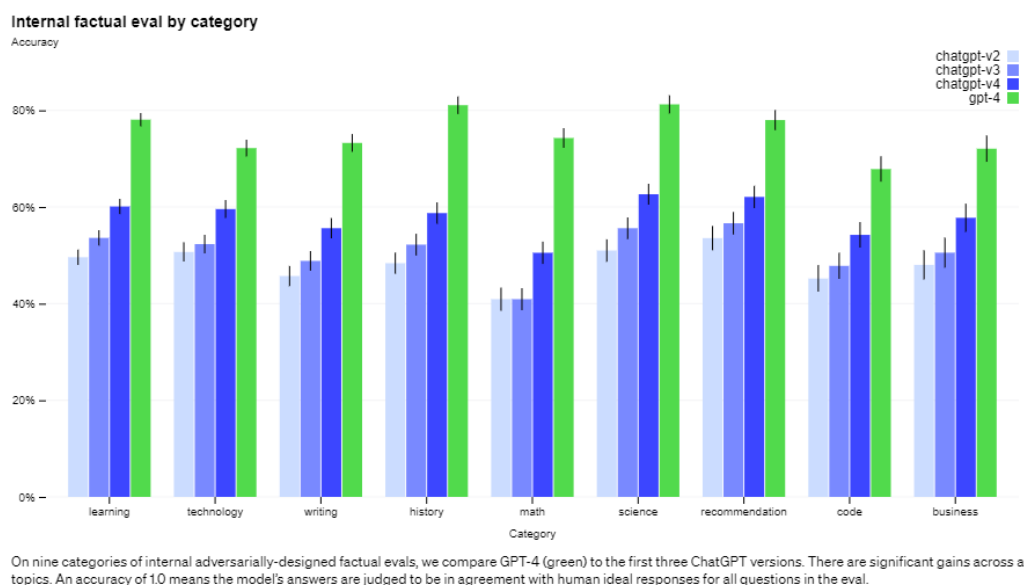
käyttäjät eivät aina pääse käyttämään, jos on paljon käyttäjiä samaan aikaan. Kirjoitushetkellä GPT-4-versio on käytössä vain maksaville asiakkaille, mutta mahdollisesti tulevaisuudessa se on kaikkien käytössä.

### ChatGPT ohjelmoijana

ChatGPT on koulutettu myös ohjelmointiin käyttäen GitHubia ja StackOwerflow'ta, joten sen avulla ohjelmointikielivalikoima on runsas ja se osaa tunnistaa lisäksi ohjelmointikieliä, vaikkakin monenlaisin tuloksin. Lisäksi monen ohjelmointikielen dokumentaatiota on osana koulutusmateriaalia, jolloin se osaa tuottaa niistä paremmin toimivaa koodia. Kielimalleja on monenlaisia, kuten GitHub Copilot, joka perustuu GPT 3 -versioon ja ehdottaa koodia suoraan ohjelmointiympäristössä. GitHub Copilot ei ole ilmainen, ja henkilökohtainen versio maksaa 10 dollaria kuussa ja yrityksille 19 dollaria jokaista käyttäjää kohti.

Insinööriyössä käytettiin ChatGPT:tä ohjeistamaan kehitystyötä, joka tehtiin Unity-pelimootorissa, jonka ohjelmointikieli on C#. ChatGPT osaa kirjoittaa C#-ohjelmointikieltä ja ymmärtää Unity-pelimootorin ominaisuudet. ChatGPT muistaa aikaisemmat kysymykset ja vastaukset, mikä helpottaa kehitystä, koska pystyy päivittämään ja lisäämään asioita. ChatGPT on ohjelmoinnissa alle 60 % oikeassa mutta versio 4 on noin 70 % oikeassa. Tämän vuoksi ohjelmoinnissa täytyy osata lukea koodia, jolloin mahdolliset virheet saadaan korjattua. (30.) ChatGPT:n oikeassa oleminen perustuu ihmisen ideaalivastaukseen vertaamiseen. Kuva 12 esittää ChatGPT:n ja GPT-4:n osaamista ohjelmoinnissa ja

muilla alueilla.



Kuva 12. ChatGPT:n ja GPT-4:n osaaminen eri alueilla, kuten ohjelmointi (30).

Virheiden löytämisen tekee vaikeaksi ihmismäinen koodi, jonka ChatGPT kirjoittaa, ja jossa on lisäksi kommentteja koodin toiminnasta. Insinöörityössä koodissa oli muutamia kertoja ylimääräisiä rivejä, jotka aiheuttivat ongelmia ja joiden löytäminen vei aikaa. Koodi vaikutti aluksi järkevältä, mutta pitkän tarkastelun, yritysten ja erehdysten kautta ongelma saatiin lopulta korjattua.

ChatGPT:llä on kehittäjille API-rajapinta, jonka avulla voi integroida tekoälyn omiin sovelluksiin. Esimerkiksi Unity-pelimoottoriin on kehitetty laajennus, jonka avulla ChatGPT voi luoda asioita suoraan pelimoottorissa (33). Tämä laajennus on tällä hetkellä vain prototyyppi, joten sen toimivuutta ei voida taata. API-rajapinta ei ole ilmainen, mutta ensimmäisenä voi ostaa noin 20 dollarin edestä poletteja, joiden arvo on noin 0,002 dollaria tuhatta polettia kohti, ja yksi poletti vastaa neljää merkkiä. Insinöörityössä ei käytetty API-rajapintaa tai Unity-laajennusta.

Koska ChatGPT on koulutettu ohjelmoimaan käyttäen GitHubia ja StackOverflow'ta. Koodi voi olla toisen ihmisen tuottamaa koodia tai lisenssin alla olevaa

koodia, jota ChatGPT tuottaa (33). Tästä ei ole saatu juridista päätöstä, mutta kannattaa olla varovainen ChatGPT:n käytössä, koska kopioinninsuojaus on käyttäjän vastuulla (33). Toinen ongelma ChatGPT:n kanssa on kysymys siitä, kuka omistaa ChatGPT:n vastaukset. OpenAI:n vastaus kysymykseen on, että käyttäjä, jonka kanssa ChatGPT keskustelee (34). Tästä tulee mahdollinen ongelma, jos kaksi henkilöä kysyy samaan ongelmaan ratkaisua tietämättään toisista ja saa identtisen vastauksen, jolloin kysymys siitä, kuka omistaa oikeudet vastaukseen, nousee esiin.

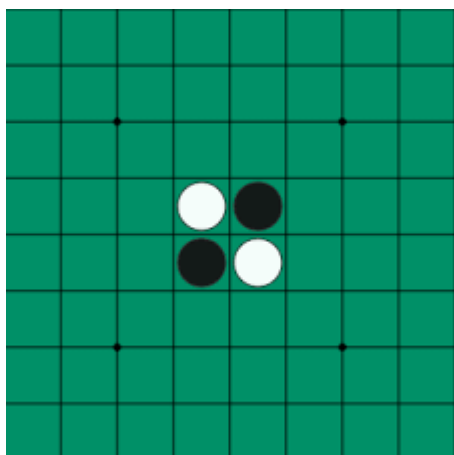
## **4 Videopelin kehittäminen tekoälyn avulla**

Insinööriyössä kehitettiin Othello-peli hyödyntäen tekoälyä kehitystyössä. Kehitys toteutettiin Unity-pelimoottorissa, joka on erinomainen pelimoottori 2D-pelin tai kevyen 3D-pelin tekemiseen. Unity valittiin pelimoottoriksi aikaisempien kokemuksien perusteella, jolloin ei tarvinnut opetella uutta pelimoottoria työn lisäksi. Ohjelmointikieleksi valittiin C#, koska se on eniten käytetty kieli kehittäjillä ja Unity-pelimoottorin pääohjelmointikieli.

### **4.1 Othello-peli**

Othello on kahden pelaajan täyden informaation nollasummapeli, jossa tavoitteena on saada mahdollisimman paljon omia nappuloita laudalle, joka on kooltaan 8 x 8. Pelissä on kaksi väriä, musta ja valkoinen, ja pelin alussa laudan

keskellä on kaksi nappulaa kummastakin väristä. Kuva 13 havainnollistaa Othello-pelin aloitustilanteen.



Kuva 13. Othello-pelin aloitustilanne (35).

Othellossa jokaisessa siirrossa täytyy vähintään yhden vastustajan väri vaihtaa omaksi, mikä tapahtuu silloin, kun kahden oman välissä on vähintään yksi toisen pelaajan nappula kaikissa mahdollisissa suunnissa. Siirrossa vaihtuvat nappulat eivät aiheuta lisääntymisiä, jos se aiheuttaa kahden oman nappulan välille vastustajan nappulan. Kaikissa suunnissa kahden omanvärisen nappulan välillä olevat vastustajan nappulat vaihtuvat omiksi nappuloiksi.

Othello-peli päättyy silloin, kun lauta on täynnä tai kun kummallakaan pelaajalla ei ole laillista siirtoa. Peli voi päättyä silloinkin, jos kaikki nappulat ovat samanvärisiä. Othello-pelin voittaja on pelaaja, jolla on enemmän nappuloita laudalla. Jos nappuloita on laudalla saman verran, peli päättyy tasapeliin. Othellossa kulmat ovat tärkeitä, koska niitä ei voi menettää. Reunojen menettäminen on vaikeampaa kuin keskustan nappuloiden menettäminen. Pelaaja voi tehdä useamman siirron peräkkäin, jos toinen pelaaja ei voi tehdä laillista siirtoa.

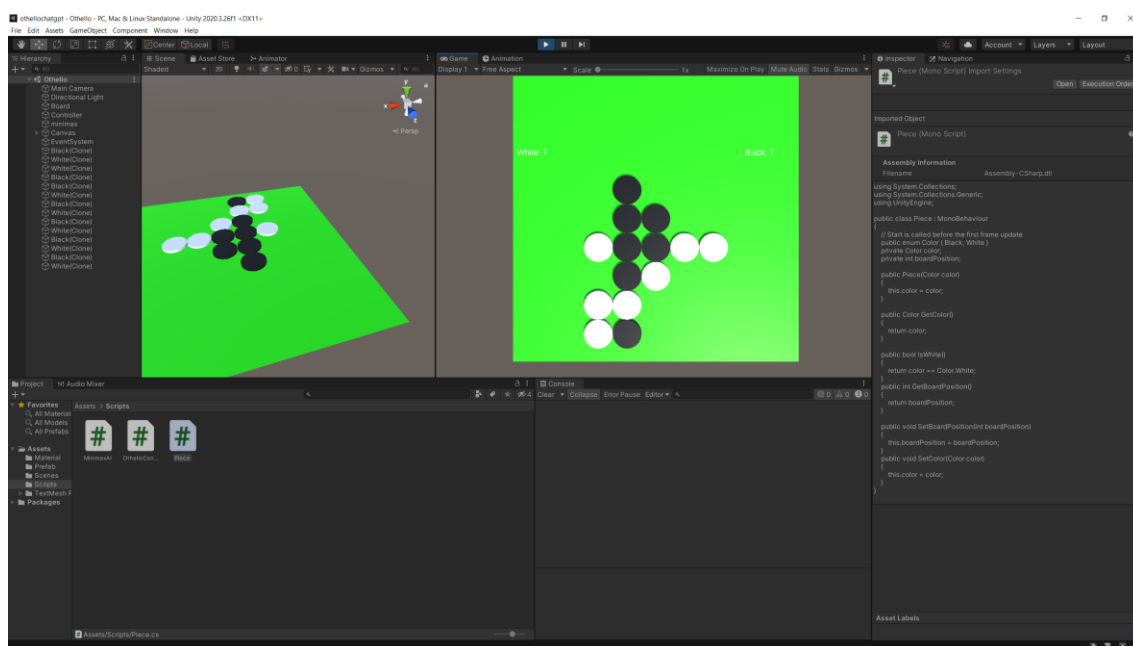
Othello-videopelin kehitysidea lähti Nintendo Entertainment Systemille vuonna 1988 julkaistusta Othello-pelistä, jossa tekoälyn paras versio voittaa monia nestistä löytyviä tekoälyjä. Miksi Othello-peliä ei ole ratkaistu, jos vuonna 1988 julkaistu peli osaa käydä pelipuuta niin pitkälle, että se voittaa viime vuosina

kehitettyjä tekoälyjä? Othello-pelin ratkaiseminen verrattuna shakkiin on helpompaa, koska Othellossa on jokaisessa siirrossa vähemmän siirtovaihtoehtoja kuin shakissa. Shakissa on useampia nappuloita, jotka siirtyvät eri tavoilla, mikä luo useita eri siirtovariaatioita.

Othello-pelistä järjestetään vuosittain suomenmestaruuskilpailuja mutta lisäksi pelataan euroopan- ja maailmanmestaruuskisoja. Tällöin vahvat tekoälyt olisivat hyväksi, sillä ne toisivat mahdollisesti uusia ideoita siirtoihin, kuten on tapahtunut shakissa tehokkaiden tekoälyjen kehittyessä.

## 4.2 Kehitystyö

Kehitystyö aloitettiin luomalla uusi projekti Unity-pelimoottorissa, ja käyttöön valittiin Unity-versio 2020.3.26f1, jolla varmistettiin pelimoottorin vakaus ja ChatGPT:n kyky toimia. ChatGPT:n viimeiset tiedot ovat syyskuulta 2021, tai ainakin käytettäessä vain tiedot rajoittuvat syyskuun 2021 asti. Kuva 14 havainnollistaa Unity-pelimoottorin käyttöliittymää, jossa Othello-peli luodaan.



Kuva 14. Unity-pelimoottorin käyttöliittymä.

Tämän jälkeen luotiin pelilauta käyttämällä plane-objektia, josta tehtiin 8 x 8 -koinen. Nappuloiden luotiin cylinder-objekteilla, joita tehtiin kahteen eri väriin. Kamera asetettiin oikealle paikalle, jotta koko pelilauta näkyisi ruudulla, mikä ei ollut automaattisesti tapahtunut.

Kun kaikki oli valmiina Unity-pelimootorissa, siirryttiin ohjelmointiympäristöön, joksi valikoitui Microsoft Visual Studio 2019 ja ohjelmointikieleksi C#. ChatGPT tuotti yli 90 % työssä käytetystä koodista. Ensimmäiseksi luotiin Othellon aloitus-tilanne, joka tapahtui siten, että luotiin nappulat laudan keskelle. Kun aloitus-tilanne oli saatu tehtyä ja testattu toimivaksi, oli aika ruveta tekemään siirtojen generointia. Aloitettiin siirtogenerointi tarkistamalla, että siirtovuorolla olevalla pelaajalla on laillisia siirtoja, minkä jälkeen tarkistettiin, onko pelaajan tekemä siirto laillinen, mistä seuraa siirron toteutus. Siirron toteuttaminen vaatii nappuloiden kääntämistä, jossa ChatGPT teki virheen, jonka vuoksi vain ensimmäinen kääntyvä nappula kääntyi, vaikka niitä olisi ollut useampia. Tämän korjaaminen vei aikansa, mutta lopulta virhe löytyi ja siirtogeneraattori oli valmis. Esimerkkikoodi 1 on esimerkki laillisten siirtojen tarkistajasta, jonka avulla varmistetaan siirron hyväksyminen, jolloin se toteutetaan.

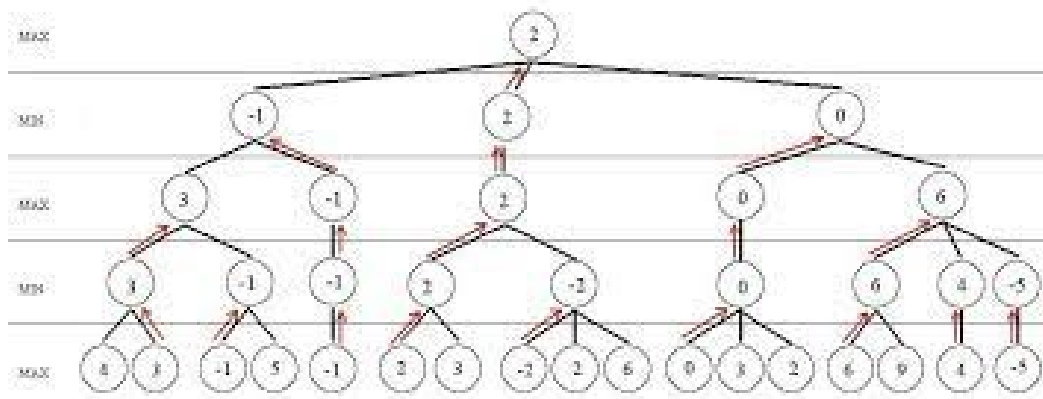
```
// Returns true if the move at (x, y) is valid
bool IsValidMove(int row, int col)
{
    // Check if the move is within the board boundaries
    if (row < 0 || row >= BOARD_SIZE || col < 0 || col >=
BOARD_SIZE)
    {
        return false;
    }
    // Check if the space is already occupied
    if (board[row, col] != 0)
    {
        return false;
    }
    // Check if the move captures any pieces
    return CheckCaptures(row, col);
}
```

Esimerkkikoodi 1. Laillisten siirtojen tarkistaja, jossa katsotaan siirron laillisuus. Jos siirto on laillinen, se hyväksytään, muuten ei hyväksytä ja siirtoa ei tapahdu.

Siirtogeneraattorin jälkeen oli aika tehdä vuorosiiro ja varmistaa, ettei peli ole päättynyt. Jos peli on päättynyt, julistetaan voittaja. Kun nämä oli tehty, oli



olemassa Othello-peli, jota pystyi pelaaman toisen ihmisen kanssa tai yksin, mutta haluttiin tehdä tekoäly, jonka kanssa pelata. Tekoälyn toteuttamiseen valittiin Minimax-algoritmi, jossa käydään pelipuuta läpi. Riippuen siitä, kenen vuoro on, minimoidaan tai maksimoidaan siirron pisteytys. Kun pelipuu on käyty läpi, tehdään paras mahdollinen siirto. Tämä tapahtuu jokaisella tekoälyn siirtovuorolla. Kuva 15 esittää Minimax-pelipuuta, jonka syvyys on 5 kerrosta mahdollisia siirtoja.



Kuva 15. Esimerkki Minimax-pelipuusta (36).

Minimax-algoritmi tarvitsee lisäksi evaluaatiofunktion, koska pelipuuta ei käydä kokonaan läpi vaan käydään tietty määrä siirtoja eteenpäin. Insinööriyössä pelipuuta käytiin 20 siirron syvyyteen, mikä tarkoittaa, että jokaista mahdollista tekoälyn siirtoa kohden käydään 20 siirtoa eteenpäin. Evaluaatiofunktio pisteyttää siirrot sen mukaan, miten ne tehdään, ja parempi evaluaatiofunktio tekee tehokkaan tekoälyn. Insinööriyössä tehty evaluaatiofunktio laskee nappuloiden määrän sen mukaan, kenen nappulat ovat kyseessä, jos omia nappuloita lisään yksi piste, ja vastustajan nappuloista vähennetään yksi piste. Tekoälyn voittaessa annetaan lisäpisteitä, ja pelaajan voittaessa vähennetään pisteitä. Lisäksi kulmista ja reunoista saa lisäpisteitä, jolloin tekoäly on vahvempi. Esimerkkikoodi 2 on esimerkki evaluaatiofunktioista, joka laskee pisteet ja palauttaa pisteet funktiolle, joka kutsuu funktiota.

```

public int EvaluateScore(int[,] board, int player)
{
    int score = 0;
    // Add or subtract the number of pieces for each player
    for (int i = 0; i < BOARD_SIZE; i++)
    {
        for (int j = 0; j < BOARD_SIZE; j++)
        {
            if (board[i, j] == player)
            {
                score++;
            }
            if (IsCorner(i, j))
            {
                // Bonus for moves that are in the corner
                score += CORNER_BONUS;
            }

            if (IsEdge(i, j))
            {
                // Bonus for moves that are on an edge
                score += EDGE_BONUS;
            }
            else if (board[i, j] == -player)
            {
                score--;
            }
        }
    }
    // Give a win bonus to the player if they have won
    if (score == BOARD_SIZE * BOARD_SIZE)
    {
        score = player == 1 ? WIN_BONUS : -WIN_BONUS;
    }
    return score;
}

```

Esimerkkikoodi 2. Evaluaatiofunktion toteutus, jossa pisteytetään siirto pelilaudan senhetkisen tilanteen mukaan.

Evaluaatiofunktio palauttaa pisteet Minimax-algoritmiin, jonka parhaat pisteet saava siirto palautetaan siirtogeneraattoriin, joka testaa sen laillisuuden ja toteuttaa laillisen siirron.

Minimax-algoritmin toteuttaminen vei eniten aikaa, koska sen toteuttaminen vaatii montaa erilaista toimivaa osaa. Yksi tällainen osa on siirtojen simulointi, jossa simuloidaan Othello-peliä, jolloin evaluaatiofunktio saa nappuloiden tapahtumat muuttamatta peliä. Minimax on rekursiivinen funktio, joka kutsuu itseään, joten täytyi olla tarkkana, että funktio kutsutaan oikeassa kohdassa, jotta kaikki mahdolliset siirrot käydään läpi. Mahdollisissa siirroissa piti olla tarkkana,

että ne olisivat laillisia siirtoja, sillä muuten tekoälyn siirrot olisivat jääneet teke-  
mättä. Esimerkkikoodi 3 esittelee koodin Minimax-algoritmin toteutuksessa,  
jossa pisteytetään kaikki mahdolliset siirrot käyttäen evaluaatiofunktia.

```

private int[] MinMax(int[,] curBoard, int depth, int player)
{
    // Get all possible moves
    List<int[]> possibleMoves = othelloController.GetPossible-
    Moves(curBoard, player);

    // If there are no possible moves, return null
    if (possibleMoves.Count == 0)
    {
        return null;
    }

    // If we have reached the maximum depth, evaluate the board
state
    if (depth == 0)
    {
        int score = othelloController.Evalu-
ateScore(curBoard,player);
        return new int[] { -1, -1, score };
    }

    int[] bestMove = new int[3];
    int bestScore = (player == OthelloController.PLAYER_1) ?
    int.MinValue : int.MaxValue;

    // For each possible move, simulate the move and recursively
    call MinMax
    foreach (int[] move in possibleMoves)
    {
        int[,] newBoard = othelloController.Make-
Move(curBoard,player, move[0],move[1]);
        int[] scoreMove = MinMax(newBoard, depth - 1, -player);

        // If we are the maximizing player, update the best score
        if (player == OthelloController.PLAYER_1)
        {
            if (scoreMove[0] > bestScore)
            {
                bestScore = scoreMove[0];
                bestMove = move;
            }
        }
        // If we are the minimizing player, update the best score
        else
        {
            if (scoreMove[0] < bestScore)
            {
                bestScore = scoreMove[0];
                bestMove = move;
            }
        }
    }

    return bestMove;
}

```

**Esimerkkikoodi 3.** Minimax-algoritmin toteutus, joka käy läpi kaikki mahdolliset siirrot ja pisteyttää ne ja palauttaa parhaan mahdollisen siirron.

Insinööriyössä tehtiin Othello-peli, jota pystyi pelaamaan tekoälyä vastaan, mutta pelin aikana puuttui tilanteen laskeminen, jolloin ei ollut helppoa nähdä, millainen tilanne pelissä oli. Tämä ongelma korjattiin lisäämällä käyttöliittymään tekstiä, jossa kerrottiin nappuloiden määrä pelilaudalla värin mukaan. Tekstin päivitystä ei tehty jokaisen ruudunpäivityksen yhteydessä, vaan se tapahtui siirtovuoron vaihtuessa, jolloin kaikki nappulat olivat päivittyneet ja nappuloiden lukumäärät olivat ajantasaisia. Esimerkkikoodi 4 on esimerkki tekstien päivittämisestä, jossa käydään läpi laudalla olevat nappulat, ja laitetaan tekstiin päivitetty pisteet.

```
public void UpdateScore()
{
    int blackScore = 0;
    int whiteScore = 0;

    // Iterate over all pieces on the board
    for (int row = 0; row < BOARD_SIZE; row++)
    {
        for (int col = 0; col < BOARD_SIZE; col++)
        {
            if (board[row, col] == 1)
            {
                blackScore++;
            }
            else if (board[row, col] == -1)
            {
                whiteScore++;
            }
        }
    }

    // Update the score display
    whiteScoreText.text = "White: " + whiteScore.ToString();
    blackScoreText.text = "Black: " + blackScore.ToString();
}
```

Esimerkkikoodi 4. Pelilaudan tuloksen päivittäminen käyttöliittymään.

### 4.3 Yhteenveto ChatGPT-tuotannosta

ChatGPT tuotti ensimmäiseksi vaiheet, joilla kehittää Othello-peli Unityssa, mikä alkoi Unity-projektin luomisesta. Kehityksessä seurattiin ChatGPT:n tuottamaa suunnitelmaa, jonka perusteella pyydettiin lisäkysymyksiä, jotka liittyivät ohjelmointiin. Ensimmäinen ohjelmointikysymys oli suoraan ChatGPT:n

vastauksesta muotoiltu pyyntö toteuttaa Othello-pelin logiikka. ChatGPT:n vastausten pituuden ylittämisen välttämiseksi se kommentoi funktioita, jolloin pyydettiin toteuttamaan funktioita. Funktioiden toteuttamisessa se teki välillä virheitä, mutta kysymällä uudestaan sai vaihtoehtoisen toteutuksen, ja näitä yhdistämällä saatiin toimiva ratkaisu.

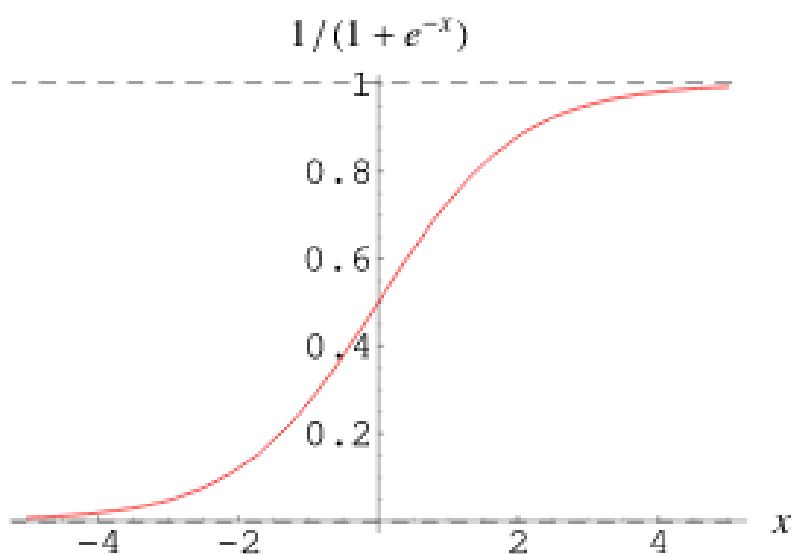
ChatGPT:n merkittävin ominaisuus ohjelmoinnin hyödyntämisessä on mahdollisuus pyytää se päivittämään sen aikaisemmin toteuttamaa koodia. Ominaisuutta hyödynnettiin nappulakoodissa, jossa nappulan väri ja sijainti laudalla säilytetään. Ensimmäiseksi luotiin nappula, joka saa värin ja palauttaa nappulan värin. Tämän jälkeen lisättiin totuusarvomuuttuja, joka selvittää nappulan värin, jonka mukaan palauttaa totuusarvon, jossa valkoinen palauttaa tosiarvon. Viimeiseksi päivitettiin nappulaan sen sijainti laudalla, jotta laudalla olevien nappuloiden tilanteen saaminen on tehokasta ja mahdollistaa pelipuun mahdollisimman syvän läpikäynnin. ChatGPT:n ominaisuus on merkittävä, koska se mahdollistaa vaikean ongelman jakamisen osiin, jolloin ratkaisu on helpompaa kuin yhden ison ongelman ratkaisu. ChatGPT osaa lisäksi kertoa muuttujan merkityksen, jos käyttäjä ei ymmärrä, mitä jokin muuttuja merkitsee.

## 5 Johtopäätökset

Insinööritö todistaa tekoälyn hyödyntämisen pelinkehityksensä oleva mahdollista ChatGPT:n kaltaisten tekoälyjen avulla, jotka osaavat tuottaa tarvittavia asioita, kuten koodia tai grafiikkaa. Esimerkiksi DALL-E-kuvanluontitekoäly luo kuvia käyttäjän syötteen perusteella, mikä on hyödyllistä pelinkehityksessä. ChatGPT:n käyttäminen pelinkehityksessä vaatii ohjelmointitaustaa, varsinkin virhetilanteissa, sillä ilman ohjelmointikokemusta ongelman löytäminen voi olla mahdotonta, vaikka ChatGPT osaakin löytää koodista virheitä ja ymmärtää virhekoodit. Pelimoottoreidenkin tuntemuksesta on hyötyä, vaikka ChatGPT ohjaa hyvin tarkasti, miten pelimoottoreissa toimitaan. ChatGPT:n käyttäminen taloudellisesti hyödyllisessä projektissa on kyseenalaista sen kopiointimahdollisuuden takia ja sen kirjoitusten omistajuuteen liittyvien epäselvyyksien takia.

Insinööritö todistaa, että tekoäly tehostaa pelin kehitystä, jolloin kehitystyö tapahtuu nopeammin eikä vaadi niin paljon ihmisiä, kun tekoäly pystyy hoitamaan aikaisemmin ihmisen hoitamia tehtäviä. Vaikka tekoäly pystyy hoitamaan monia asioita, se on kuitenkin vain työkalu, jota ihmiset käyttävät. Tekoälyt ovat yhtä hyviä kuin niiden käyttäjät, joten ihmisten täytyy oppia käyttämään niitä tehokkaasti. Insinööritöyön aikana opittiin käyttämään tekoälyä paremmin ajan myötä ja löydettiin hyviä keinoja saada tekoälyltä halutut asiat.

Tekoälyn hyödyntäminen lisääntyy tulevaisuudessa, ja sen kehitystä voi verrata puhelimen kehittymiseen, jossa kosketusnäytön jälkeen ei ole tapahtunut merkittävää kehitystä. Tätä kuvastaa logistinen funktio, jossa ajan myötä kehitys hidastuu, ja aluksi kehitys on nopeaa, kun tehdään mullistavia muutoksia teknologiaan ja mahdollisesti yhteiskuntaan (37). Kuva 16 havainnollistaa logistista funktiota.



Kuva 16. Logistisen funktion kuvaaja (38).

Kysymys kuuluu, missä kohtaa kuvaajaa ollaan tekoälyn kehityksen suhteen. Jos ollaan kehityksen alkuvaiheessa, tekoälyn toiminta paranee ja muutokset voivat olla teollisen vallankumouksen kaltaisia. Toisaalta, jos kehitys on jo päässyt huippuunsa, kuinka mullistava tekoäly on jo nykyisellään?

Tekoälyjen hyödyntäminen ohjelmointityössä voi olla tulevaisuudessa lähtökohta, jonka jälkeen vasta oma ohjelmointiosaaminen tulee tärkeysjärjestyksessä. Tekoäly voi pahimmillaan viedä työpaikkoja aloilta, jotka voidaan automatisoida, esimerkiksi tiedonsyöttö, toistuvat hallintotehtävät ja kuljetus (39). Tekoälyissä saattaa olla tietosuojaongelmia, sillä Italiassa ChatGPT on kielletty siihen asti, kunnes OpenAI vastaa Italian tietosuojaan liittyviin kysymyksiin (40).



## Lähteet

- 1 Aaltonen, Mika & Merilehto, Antti. 2019. Tekoäly: ihminen ja kone. E-kirja. Helsinki: Alma Talent.
- 2 Schroer, Alyssa. 2022. Artificial Intelligence. What Is Artificial Intelligence (AI)? How Does AI Work? Verkkoaineisto. Built in. <<https://builtin.com/artificial-intelligence>>. Päivitetty 3.3.2023. Luettu 23.4.2023.
- 3 Deep Blue. Verkkoaineisto. IBM. <<https://www.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/>>. Luettu 23.4.2023.
- 4 Goodrow, Cristos. 2021. On YouTube's recommendation system. Verkkoaineisto. Youtube. <<https://blog.youtube/inside-youtube/on-youtubes-recommendation-system/>>. Luettu 23.4.2023.
- 5 Raatikainen, Panu (toim.). 2021. Tekoäly, ihminen ja yhteiskunta: filosofisia näkökulmia. E-kirja. Gaudeamus.
- 6 Hands-on Game AI Development: What Is a Finite-State Machine? 2018. Verkkoaineisto. Youtube. <<https://www.youtube.com/watch?v=3AVC4sO4Tx4>>. Luettu 28.4.2023.
- 7 What is machine learning? Verkkoaineisto. IBM. <<https://www.ibm.com/topics/machine-learning>>. Luettu 23.4.2023.
- 8 TensorFlow. Verkkoaineisto. TensorFlow. <<https://www.tensorflow.org/>>. Luettu 23.4.2023.
- 9 PyTorch. Verkkoaineisto. PyTorch. <<https://pytorch.org/>>. Luettu 23.4.2023.
- 10 Yegulalp, Serdar. 2022. What is TensorFlow? The machine learning library explained. Verkkoaineisto. InfoWorld. <<https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html>>. Luettu 23.4.2023.
- 11 What's coming in TensorFlow 2.0. 2019. Verkkoaineisto. TensorFlow. <<https://blog.tensorflow.org/2019/01/whats-coming-in-tensorflow-2-0.html>>. Luettu 28.4.2023.
- 12 What is PyTorch, and How Does It Work: All You Need to Know. 2023. Verkkoaineisto. Simplilearn. <<https://www.simplilearn.com/what-is-pytorch-article>>. Luettu 23.4.2023.

- 13 01. PyTorch Workflow Fundamentals. Verkkoaineisto. Learnpytorch. <[https://www.learnpytorch.io/01\\_pytorch\\_workflow/](https://www.learnpytorch.io/01_pytorch_workflow/)>. Luettu 28.4.2023.
- 14 What Is A Neural Network? Verkkoaineisto. Amazon. <<https://aws.amazon.com/what-is/neural-network/>>. Luettu 23.4.2023.
- 15 What are neural networks? Verkkoaineisto. IBM. <<https://www.ibm.com/topics/neural-networks>>. Luettu 29.4.2023.
- 16 Tuominen, Heli & Neittaanmäki, Pekka (toim.). 2019. Tekoälyn perusteita ja sovelluksia. E-kirja. Jyväskylän yliopisto.
- 17 What is deep learning? Verkkoaineisto. IBM. <<https://www.ibm.com/topics/deep-learning>>. Luettu 23.4.2023.
- 18 Salatas John. 2011. Implementation of Elman Recurrent Neural Network in WEKA. Verkkoaineisto. ICT Research Blog. <<https://jsalatas.ictpro.gr/implementation-of-elman-recurrent-neural-network-in-weka/>>. Luettu 28.4.2023.
- 19 Convolutional Neural Networks. Verkkoaineisto. IBM. <<https://www.ibm.com/topics/convolutional-neural-networks>>. Luettu 24.4.2023.
- 20 Ruokolainen, Lasse. 2018. Ylisovittaminen-ja-kuinka-sen-kanssa-voi-tulla-toimeen. Verkkoaineisto. Bilot. <<https://bilot.group/articles/ylisovittaminen-ja-kuinka-sen-kanssa-voi-tulla-toimeen/>>. Luettu 24.4.2023.
- 21 Mandal, Manav. 2021. Introduction to Convolutional Neural Networks (CNN). Verkkoaineisto. Analytics Vidhya. <<https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/>>. Päivitetty 28.4.2023. Luettu 28.4.2023.
- 22 What are recurrent neural networks? Verkkoaineisto. IBM. <<https://www.ibm.com/topics/recurrent-neural-networks>>. Luettu 24.4.2023.
- 23 Introduction to Recurrent Neural Network. Verkkoaineisto. Geeksforgeeks. <<https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/>>. Luettu 24.4.2023.
- 24 Uszkoreit, Jakob. 2017. Transformer: A Novel Neural Network Architecture for Language Understanding. Verkkoaineisto. Google. <<https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html>>. Luettu 24.4.2023.

- 25 Merritt, Rick. 2022. What Is a Transformer Model? Verkkoaineisto. Nvidia. <<https://blogs.nvidia.com/blog/2022/03/25/what-is-a-transformer-model/>>. Luettu 24.4.2023.
- 26 ChatGPT. Verkkoaineisto. OpenAI. <<https://chat.openai.com/>>. Luettu 10.1.2023.
- 27 Christensen, Alexander. 2023. How Many Languages Does ChatGPT Support? Verkkoaineisto. Seo. <<https://seo.ai/blog/how-many-languages-does-chatgpt-support>>. Luettu 25.4.2023.
- 28 Introducing ChatGPT. 2022. Verkkoaineisto. OpenAI. <<https://openai.com/blog/chatgpt>>. Luettu 10.1.2023.
- 29 Loukides, Mike. 2023. What Are ChatGPT and Its Friends? E-kirja. O'Reilly.
- 30 Gpt-4 technical report. 2023. Verkkoaineisto. OpenAI. <<https://arxiv.org/pdf/2303.08774.pdf>>. Päivitetty 27.3.2023. Luettu 25.4.2023.
- 31 Lindström, Suvi. 2020. Tekoälytutkimuksen klassikko Turingin testi mittaa tietokoneen älykkyyttä – sen kehittäjä Alan Turing koki synkän kohtalon. Verkkoaineisto. itewiki. <<https://www.itewiki.fi/blog/2020/08/tekoalytutkimuksen-klassikko-turingin-testi-mittaa-tietokoneen-alykkyytta-sen-kehittaja-alan-turing-koki-synkan-kohtalon/>>. Luettu 25.4.2023.
- 32 OpenAI's ChatGPT Reportedly Costs \$100,000 a Day to Run. Verkkoaineisto. CIOCoverage. <<https://www.ciocoverage.com/openais-chatgpt-reportedly-costs-100000-a-day-to-run/>>. Luettu 25.4.2023.
- 33 ChatGPT in Unity is INSANE! 2023. Verkkoaineisto. Youtube. <<https://www.youtube.com/watch?v=1WnYAYbytVU>>. Luettu 25.4.2023.
- 34 Morgan, Paul. 2023. The good, the bad and the ugly: ChatGPT and beyond. Verkkoaineisto. The Media Online. <<https://themediainline.co.za/2023/03/the-good-the-bad-and-the-ugly-chatgpt-and-beyond/>>. Luettu 25.4.2023.
- 35 eOthello. Verkkoaineisto. eOthello. <<https://www.eothello.com/>>. Luettu 28.4.2023.
- 36 Karhunen, Jaakko. 2018. Minimax ja alfa-beta-karsinta. Kandidaatin tutkielma. Jyväskylän yliopisto. Jyx-tietokanta.

- 37 Scott, Tom. 2023. I tried using AI. It scared me. Verkkoaineisto. Youtube. <<https://www.youtube.com/watch?v=jPhJbKBuNnA>>. Luettu 27.4.2023.
- 38 Sigmoid function. Verkkoaineisto. Wolfram Mathworld. <<https://math-world.wolfram.com/SigmoidFunction.html>>. Luettu 28.4.2023.
- 39 Koskinen, Petri. 2023. Tekoäly: Näiltä aloilta vien ensimmäisenä työpaikat – asiantuntijat uskovat samaan. Verkkoaineisto. Tivi. <<https://www.tivi.fi/uutiset/tekoaly-nailta-aloilta-vien-ensimmaisena-tyopaikat-asiantuntijat-uskovat-samaan/d1bc861c-cb6f-4aa2-97d7-012cf3cf221a>>. Luettu 27.4.2023.
- 40 Heleskoski, Janne. 2023. ChatGPT kiellettiin jo yhdessä EU-maassa, ja se saattoi olla vasta alkusoittoa – tällaisia muutoksia vaaditaan. Verkkoaineisto. Tivi. <<https://www.tivi.fi/uutiset/chatgpt-kiellettiin-jo-yhdessa-eu-maassa-ja-se-saattoi-olla-vasta-alkusoittoa-tallaisia-muutoksia-vaaditaan/975b90a8-7c80-4890-88cc-0c83a2df13ae>>. Luettu 27.4.2023.