LSTM-model forecast af aktiekurser til porteføljesammensætning

Aalborg Universitet

Asger Ahlmann-Laursen _____

Steffen Suwan Christensen _____

Jonas Høst Mazur _____

Vejleder: Lasse Bork

10. Semester Oecon 2021

Abstract

We set up multiple LSTM networks to predict the weekly returns of ten different stocks during the uncertainty of the Covid-19 pandemic. Three different models were proposed for each stock, which were trained on historic pricing data of different frequencies and sequence lengths. Two of these models used 5-minute price data in sequences of 3 and 5 trading days, respectively. The third model used daily closing price observations in sequences of 90 trading days.

Out-of-sample returns were forecasted to evaluate the economic value of the LSTM models. These were then used in an MVO framework to construct optimal portfolios based on a risk averse investor. A simple DCC-GARCH model was created to forecast the variance-covariance matrix of the ten stocks. All the forecasted returns and variance-covariance matrices were based on out-of-sample test data, ensuring that the networks had not seen the data. We do not find any significant economic gain by using the LSTM based forecast in constructing portfolios compared to a naive forecast during the out-of-sample period.

Indholdsfortegnelse

1 Introduktion **1**

2 Data **4**

3 Teori **5**

3.1 Markowitz Mean-variance 5

3.2 Økonomisk evaluering af porteføljer 8

3.3 Volatilitetsmodellering 9

3.3.1 GARCH 10

3.3.2 DCC-GARCH 11

3.3.3 Estimation og forecasting 14

3.3.4 Ljung-Box Test 15

3.4 Machine Learning 16

3.4.1 Artificial Neural Network 16

3.4.2 Recurrent Neural Network 23

4 Modellering **28**

4.1 Model Arkitektur 29

4.1.1 Inputsdata 31

4.1.2 Univariate GARCH og DCC-GARCH 33

4.2 Model resultater 34

4.3 Investeringsstrategi 37

4.3.1 Økonomisk evaluering 39

5 Diskussion **40**

5.1 Kritik af Mean-Variance frameworket 40

5.2 Machine Learning og finans 42

5.2.1 Lav signal-to-noise ratio 42

5.2.2 Kapitalforvaltning og frekvens 43

5.2.3 Inkorporering af økonomisk teori 44

5.2.4 Netværksarkitektur - mere kunst end videnskab 44

5.2.5 Reproducerbarhed og fortolkning af kunstige neurale netværk 46

6 Konklusion **47**

7 Litteraturliste **48**

1. Introduktion

Investeringssammensætning er et centralt emne i finansiel teori, hvoraf forventningen til afkastet af en investering er basal for emnet. Derfor har der både historisk og fortsat i dag gået meget arbejde i at udarbejde metoder til at forudsige fremtidigt afkast for investeringer og tilhørende, hvordan man optimalt sammensætter en investeringsstrategi baseret på en investors præferencer og forventninger. Blandt disse metoder har økonometriske eller statistiske metoder, baseret på analyse af historiske markedsbevægelser, været de mest adopterede og anvendte i feltet (Atsalakis & Valavanis, 2013). Udfordringen med disse metoder er, at opbygningen af modellerne er baseret på en række antagelser samt teori om, hvad der bestemmer aktiemarkedsudviklingen og hvordan. Antallet af variable, som tidligere arbejde inden for feltet påpeger at have prædiktive signaler, tælles i de hundredevis (se f.eks. Green et. al, 2013 eller Harvey et al., 2016). Men traditionelle metoder holder ikke op, når antallet af variable er oppe i så højt et antal, og desuden er der høj korrelation imellem de prædiktive faktorer. Valget om, hvilke variable og hvor stort et antal, der skal anvendes i modelleringen, og om disse overhovedet direkte kan observeres, er ikke uden udfordringer. Der er også empirisk problematik i, at variable, der udemærket forklarer historisk observerede markedsudviklinger, falder fra hinanden, når det gælder forudsigelse af fremtidigt afkast. Relevansen af disse variable, samt den information og de muligvis komplekse ikke-lineære forhold, disse måtte have med markedsudviklingen, er desuden sjældent velspecificeret af finansiel økonomisk teori (Heaton et. al. 2016).

De teknologiske fremskridt opnået indenfor machine learning har åbnet op for nye udsigter som et alternativ til de traditionelle økonometriske modeller. Udviklingen af nye læringsalgoritmer og øget computationel efficiens har resulteret i gode resultater i modelleringen af ikke-lineære og komplekse sammenhænge[1]. Anvendelsen af kunstige neurale netværk har været attraktiv i den finansielle verden på grund af udviklingen af metoder inden for machine learnings evne til at reducere dimensionaliteten af data og forklarende variable, samt muligheden for ikke-lineære interaktioner med de forklarende variable[2]. Deep learning modeller har evnen til at uddrage abstrakte egenskaber fra data og identificere skjulte forhold, uden at modelarkitekten på forhånd har taget en række økonometriske og teoretiske antagelser. Dette gør disse modeltyper særligt attraktive inden for aktiemarkeds forecasting, netop fordi

[1] Se Dargan et. al (2019) for en bred gennemgang af Deep learnings egenskaber, succeser og anvendelse over mange felter.

[2] Se f.eks. Hinton & Salakhutdinov (2006)

teorien inden for feltet peger på så mange forklarende variable, som bør indeholde information, der påvirker aktiemarkedet. Endnu en fordel ved machine learning-tilgangen er, at økonometriske udfordringer, som f.eks. kollinearitet i variablene eller heteroskedicitet i forecastingsmodellen, er af mindre eller ingen betydning for modelleringen. Som Breiman et. al. (2001) pointerer, er fokusset i denne tilgang at maksimere nøjagtigheden af forudsigelserne på en ukendt datadreven proces, hvorimod den traditionelle, statistiske tilgang er at estimere en i forvejen kendt eller teoretisk baseret proces og udarbejde hypotesetest af denne efterfølgende[3]. Anvendelsen af kunstige neurale netværk i litteraturen finder lovende resultater af disse modellers evne til at forudsige aktiepriser og opnår ofte bedre resultater end de konventionelle metoder (Atsalakis & Valavanis, 2009; Gu, Kelly & Xiu, 2019).
Tilgængelighed af data og nyheder har været i fokus de sidste mange år og bliver højst sandsynligt kun bedre herfra. Konstant og let adgang til den nyeste informationstrøm medfører,, at reaktionerne på markedet kan ændre sig fra minut til minut. Dette betyder også, at muligheden for at analysere på denne datastrøm er til stede og bør udforskes. Som Chong et. al (2017) argumenterer for anvendelsen af neurale netværk: *"Its ability to extract features from a large set of raw data without relying on prior knowledge of predictors makes deep learning potentially attractive for stock market prediction at high frequencies."*. Høje frekvenser er et vigtigt term fra dette citat. Som Kelly et. al (2020) argumenterer, er det korrekt, at disse nye machine learning metoder kan håndtere et væld af variable samtidigt, men antallet af observationer, som modellerne skal lære fra, er den vigtigste faktor for succesen af machine learning inden for mange felter. Emnet om modellering af aktieafkast har hidtil været et såkaldt *small data* problem[4]. Af flere årsager[5] har dataet anvendt inden for feltet hidtil været domineret af daglige, ugentlige eller månedlige prisobservationer. Kelly et. al. (2020) argumenterer, at på sådanne frekvensniveau vil antallet af tilgængelige observationer ikke være tilstrækkeligt for et kunstigt neuralt netværk at lære af. Hvorfor ikke anvende de konventionelle økonometriske metoder på intradags data f.eks. minut til minut? Traditionelle metoder er potentielt ikke velegnede til anvendelse af høj-frekvens data, fordi nogle af de økonomiske effekter, teorien påpeger, kan forsvinde inter-temporalt. I disse metoder vælges faktorer og forklarende variable bevidst som regel ud fra teori, men effekter, der er

[3]Det er diskutabelt og fortsat diskuteret i litteraturen, om dette bør anses for en fordel. Et argument imod er, at disse modeller kan sammenlignes med en sort boks, hvor det er umuligt at tolke på egenskaberne af modellen og de sammenhænge, som den måtte finde.

[4]Betydende, at der ikke er et tilstrækkeligt antal dataobservationer.

[5]Den åbenlyse årsag er tilgængeligheden af data. Andre årsager kan være af teoretiske eller praktiske årsager. F.eks., at der tages højde for handelsomkostninger, hvor ofte investeringsbeslutninger tages osv.

gældende på ugeplan, kan potentielt forsvinde, når der analyseres på minut til minut udviklinger. Der er andet research, som har anvendt høj-frekvens aktiedata i neurale netværk (se f.eks. Arévalo et. al, 2016, Chong et. al, 2017) men få med de nyere netværksstrukturer, som er udviklet og hyppigt brugt inden for andre brancher. Freitas et. al (2009) anvender f.eks. en Long Short-Term Memory model på ugentligt data af det brasilianske IBOVESPA aktieindeks og sammensætter porteføljer baserede på et mean-variance optimization (MVO) framework men bruger et meget simpelt mål for risiko og kovarians imellem aktiverne.
I dette projekt vil der blive konstrueret deep learning Recurrent Neural Network i form af Long Short-Term Memory modeller til forecasting af den ugentlige prisudvikling for en række aktier trænet på høj-frekvens intradags trading-data. Både in-sample og out-of-sample dataet, anvendt til at træne og evaluere forecastene, vil foregå i tidsperioden af den globale Covid-19 pandemi. Interessen er her at se, om anvendelsen af neurale netværk kan formå at fange den høje volatilitet, som aktiemarkederne oplevede og fortsat oplever i denne tidsperiode. Er det muligt at fange sådan volatilitet i en forecastingsmodel, vil der være belæg for at føre en mere aktiv porteføljeforvaltning i perioden for at minimere tab og maksimere afkast. Forecasting-modellerne vil efterfølgende blive anvendt til sammensætningen af en "optimal" investeringsportefølje via et MVO-framework. Dette er valgt, da det giver en regel-baseret investeringsstrategi, som i teorien minimerer risiko til et givet, forventet afkast for en portefølje af korrelerede aktiver. På denne måde bliver præstationen af forecastingsmodellerne ikke blot evalueret på deres fejl-estimater men også pragmatisk testet i en out-of-sample investeringsstrategi. Porteføljesammensætningen re-allokeres på ugentlig basis for at holde frekvensen af handelsomkostninger nede[6]. Til porteføljekonstruktion er det relevant at anskue mange forskellige aktier, hvorfor det er fordelagtigt at anvende DCC-GARCH til estimering af kovariansen imellem aktiverne i investeringsporteføljen. DCC-modellen lider ikke af dimensionalitetsproblemer, modsat mange andre multivariate GARCH-modeller, og modellen har derfor nemmere ved at estimere store kovarians matricer. For at evaluere den økonomiske værdi af de estimerede forecast, vil en porteføljeafgift blive udregnet, som sætter den økonomiske nytte af de sammensatte porteføljer baseret på vores modelafkast lig nytten fra naive forecast, i samme stil som Fleming et al. (2001).
Det er vigtigt at pointere, at formålet med forecastingsdelen ikke vil være at direkte påpege teoretisk og økonomisk sammenhæng mellem variablene anvendt i træningen af det neurale netværk. Ligeledes er det ikke målet, at teste hvorvidt, eller i hvilken grad, variablene indeholder information, der er gældende for kursudviklingen. I stedet skal det ses som

[6]Der vælges dog at ses bort fra transaktionsomkostninger.

en undersøgelse og evaluering af en kunstig neural netværks-strukturs, der er trænet på højfrekvens finansielt data, evne til at opnå forecast som har en økonomisk værdi under en tidsperiode med meget høj markedsvolatilitet.
Projektet er bygget op som følgende. Sektion 2 vil indeholde en gennemgang af det anvendte data, herunder de aktier, som er blevet inddraget. Teorien bag MVO-frameworket og DCC-GARCH vil kort blive præsenteret i sektion 3, hvor der desuden vil være en dybdegående teoretisk gennemgang af kunstige neurale netværk. I sektion 4 opsættes de forskellige modeller, der anvendes til forecasting af aktieafkast og kovarians matricer. Herefter vil resultaterne af modellerne blive evaluerede ved at sammensætte porteføljer baserede på modellernes forecast. Problemstillingerne og overvejelser, der bør tages højde for vedrørende modellering af aktieafkast via machine learning teknikker, vil blive bragt op og diskuterede i sektion 5. Afslutningsvis vil der i sektion 6 blive konkluderet på dette projekts resultater.

2. Data

Projektets datasæt består af 10 aktier, der har været del af Dow Jones Industrial Average indekset siden før år 2000. Dow Jones indekset består af 30 large-cap aktier fra amerikanske børser. Ved at anvende aktier, der har indgået i indekset i over 20 år, vil der findes modne large-cap virksomheder, hvor der generelt skal en betydelig mængde kapital til at flytte aktiernes priser. Aktierne vil også have et smallere bid-ask spread sammenlignet med mindre aktier, eftersom virksomhederne følges af flere aktører. Dette reducerer sandsynligheden for prisanomalier, hvilke ville have en forstyrrende effekt i forbindelse med modelleringen. Blandt de 16 aktier, der har indgået i indekset i over 20 år, søges at sammensætte porteføljer af 10 aktier, således porteføljerne eksponeres overfor flest mulige sektorer og tilhørende risikofaktorer. Porteføljerne består af følgende aktier og sektorer:

- *The Procter & Gamble Company* ($PG) - Forbrugsdefensive sektor.
- *Walmart Inc.* ($WMT) - Forbrugsdefensive sektor.
- *McDonald's Corporation* ($MCD) - Forbrugscykliske sektor.
- *The Boeing Company* ($BA) - Industrisektor.
- *Caterpillar Inc.* ($CAT) - Industrisektor.
- *Microsoft Corporation* ($MSFT) - Teknologisektor.
- *International Business Machines Corporation* ($IBM) - Teknologisektor.

- *Merck & Co., Inc.* ($MRK) - Sundhedssektor.
- *JPMorgan Chase & Co.* ($JPM) - Finansielle sektor.
- *The Walt Disney Company* ($DIS) - Communication service sektor.

Porteføljerne kan herved eksponeres overfor maksimalt syv sektorer. Mange af disse sektorer har fælles risikofaktorer, men generelt vil inkludering af flere sektorer medføre, at idiosynkratisk risiko har mindre indflydelse på porteføljens udvikling.
Til projektets porteføljesammensætning anvendes i alt tre datasæt af forskellige tidsfrekvenser. Det ene datasæt består af ugentlige prisændringer i procent, fra onsdag til onsdag, siden år 2010, hvilket anvendes til volatilitetsmodellering. Det andet datasæt består af intradags dataændringer i aktiekurserne, hvor priserne opdateres hvert femte minut, hvilket anvendes af projektets machine learning modeller. Det var kun muligt at anskaffe to års "højfrekvens" data, hvorfor datasættet starter den 8. maj 2019 og slutter den 21. april 2021. Originalt indeholdt dette data også prisinformation udenfor den normale åbningstid, 9:30 til 16:00, hvor der ikke nødvendigvis er ny data hvert femte minut for alle aktier. Dette fravælges for at have mere homogene tidssekvenser og yderligere sikre kvalitet af prisinformationen, eftersom likviditeten mindskes signifikant i for- og eftermarkedshandel. For at teste projektets machine learning modeller med datasæt af flere observationer, anvendes et tredje datasæt bestående af dagligt data indsamlet siden år 2000. Ved at der undersøges en længere tidsperiode, forøges sandsynligheden for at signaler, der findes i træningssættet, ikke længere er relevante i testperioden. Modeller til brug i praksis bør af denne grund også hyppigt revideres med ny data.

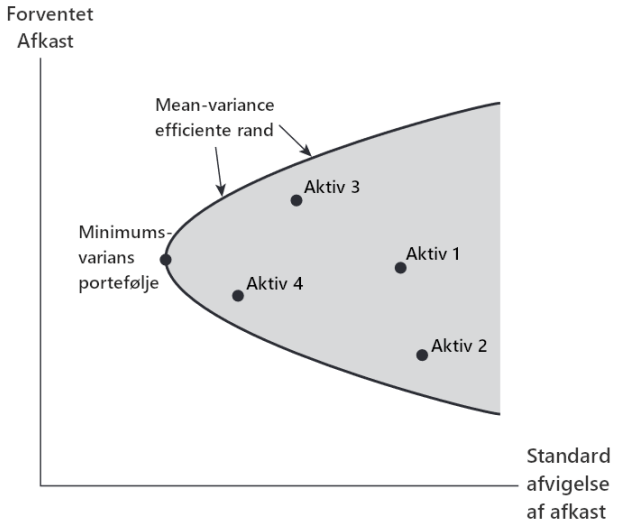
3. Teori

3.1 Markowitz Mean-variance

Den ideelle porteføljesammensætning implementerer både de teoretisk relevante forhold for investoren og investeringsaktiverne og finder en løsning til dette problem via afkastdataet. En af de mest kendte og anvendte teorier inden for investeringsporteføljer er Harry Markowitz' mean-variance framework (Markowitz, 1952).
Mean-variance teorien argumenterer, at en investor ikke alene bør overveje det forventede afkast af en investering men også inkludere et mål for risiko. Fordi afkastet i fremtiden ikke er kendt på forhånd, må investeringsbeslutningerne for en investor være baserede på

forventningen om det fremtidige afkast af et aktiv. Men Markowitz påpeger, at det ikke kun er det forventede afkast, som en investor bør overveje. Teorien understreger, at investoren også bør tage højde for risikoen af investeringen i og med, at det fremtidige afkast ikke er kendt på forhånd og indebærer derfor en vis usikkerhed. Risikomålet bør ikke kun vurderes ud fra risikoen ved investeringen for det enkelte aktiv, men bør også relatere denne risiko til hele investeringsporteføljen. Værdien af finansielle aktiver og udviklingen af disse kovarierer som regel med hinanden, og dette skal der tages højde for, hvis der korrekt skal diversificeres for den overordnede risiko af en portefølje bestående af kovarierende, risikofyldte aktiver. Markowitz anvender i frameworket variansen af de enkelte aktivers afkast som risikomål.
Udover teoretisk at fremlægge, hvad en investor bør tage højde for i sin investeringbeslutning, giver mean-variance frameworket også en analytisk løsning på, hvordan en optimal sammensætning af risikofyldte aktiver samles. I sin teori anser Markowitz det fremtidige afkast som en stokastisk variabel og kan derfra anvende en række egenskaber fra statistik, som en investor bør tage med i sin investeringssammensætning. Givet en investors forventninger til de fremtidige afkast og variansen af disse, står investoren overfor en beslutning om sammensætningen af en investeringsportefølje, hvoraf Markowitz påpeger, at der findes en række "optimale" porteføljesammensætningner alt efter investorens præferencer.
Markowitz' mean-variance paradigme medfører interessante resultater. Paradigmet medfører, at der for enhver samling af aktiver kan sammensættes en dominerende portefølje baseret på afkast og risiko. Givet en efficient porteføljesammensætning, vil det kun være muligt at opnå et højere afkast ved at påtage sig større risiko. Disse egenskaber er illustrerede i det kendte mean-variance diagram, som kan ses i figur 3.1.

Fig. 3.1: Mean-Variance Diagram



Diagrammet illustrerer netop, at der vil være specifikke porteføljer som *dominerer* andre. En dominant portefølje vil være en portefølje, som enten har et højere forventet afkast til samme risiko, eller er mindre risikabel, men har det samme forventede afkast sammenlignet med en anden portefølje. De dominerende eller efficiente porteføljer er i diagrammet repræsenterede i den øvre rand af det hyperbole område[1], der består af alle de mulige porteføljesammensætninger af de aktiver, som inkluderes i investeringsbeslutningen.
Under mean-variance paradigmet er det muligt at løse for disse efficiente porteføljer med blot tre informationssæt. Givet N aktiver kræver MVO blot det forventede afkast og varians af alle aktiver, samt kovariansen mellem aktiverne. Det forventede afkast af en portefølje bestemmes som følgende

$$\mu_P = E(\tilde{r}_P) = \mathbf{w}^\top \boldsymbol{\mu} \tag{3.1}$$

hvor $\boldsymbol{\mu}$ og $\mathbf{w}$ repræsenterer vektorer af henholdsvis de enkelte aktivers forventede afkast og vægte i porteføljen:

$$\boldsymbol{\mu} = \begin{pmatrix} E[\tilde{r}_1] \\ E[\tilde{r}_2] \\ \vdots \\ E[\tilde{r}_N] \end{pmatrix} \qquad \mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix} \tag{3.2}$$

Variansen af en given portefølje kan findes ved

$$\sigma_P^2 = \mathbf{w}^\top \boldsymbol{\Sigma} \mathbf{w} \tag{3.3}$$

hvor $\boldsymbol{\Sigma}$ er en $N \times N$ varians-kovarians matrice mellem afkastene på de N aktiver, som følgende

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1N} \\ \sigma_{12} & \sigma_2^2 & \cdots & \sigma_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{1N} & \sigma_{2N} & \cdots & \sigma_N^2 \end{bmatrix} \tag{3.4}$$

Beslutningen en investor står overfor i mean-variance paradigmet, vil være at sammensætte vægtene på de N risikofyldte aktiver $\mathbf{w}$, så porteføljen opfylder investorens præferencer mellem afkast og risiko. I mean-variance frameworket overvejer investoren et kvadratisk nyttemål, tilsvarende forskellen imellem variansen (ganget med en parameter γ, der repræsenterer investorens risikoaversitet) og det forventede afkast af en portefølje. En mean-variance investor kan da vælge sin optimale portefølje alt efter, hvor villig eller avers investoren måtte

[1] Den hyperbole form skyldes, at aktiver sjældent er perfekt korrelerede.

være overfor risiko. Problemstillingen, som investoren står overfor, kan repræsenteres som i følgende:

$$\min_{\mathbf{w}} \frac{\gamma}{2}\mathbf{w}^\top \boldsymbol{\Sigma}\mathbf{w} - \mathbf{w}^\top \boldsymbol{\mu} \text{ s.t. } \sum_{i=1}^{N} w_i = \mathbf{w}^\top \mathbf{1} = 1 \tag{3.5}$$

hvor γ repræsenterer en risikoaversitets-parameter. Optimeringsproblemet er todelt, hvor det første led, $\frac{\gamma}{2}\mathbf{w}^\top \boldsymbol{\Sigma}\mathbf{w}$, repræsenterer risikomålet eller variansen af en portefølje og det andet led, $\mathbf{w}^\top \boldsymbol{\mu}$, er det forventede afkast som i ligning (3.3) og (3.1). Jo mere risikoavers en investor, desto større en parameterværdi vil γ tage, og variansen af investorens foretrukne portefølje bør da mindskes for at opfylde dennes præferencer. Det kan vises, at minimeringsproblemet i ligning (3.5) har en unik løsning for enhver investors risikoaversitets-parameter γ og given portefølje af aktiver, som følgende[2]

$$\mathbf{w}^* = \frac{A}{\gamma}\frac{\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}}{A} + \left(1 - \frac{A}{\gamma}\right)\frac{\boldsymbol{\Sigma}^{-1}\mathbf{1}}{C} \tag{3.6}$$

hvor $\mathbf{1}$ er en $N \times 1$ vektor af 1-taller, $A = \mathbf{1}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ og $C = \mathbf{1}^\top \boldsymbol{\Sigma}^{-1}\mathbf{1}$.
Porteføljesammensætning via mean-variance frameworket er i dette projekt valgt, fordi det giver en regelbaseret investeringsstrategi med en analytisk løsning, baseret på forventet afkast samt risikomål.

3.2 Økonomisk evaluering af porteføljer

MVO-frameworket vil i projektet blive anvendt til at sammensætte porteføljer baserede på afkast forecasts via kunstige neurale netværk. For at vurdere om de sammensatte porteføljer, baserede på forventningerne fra disse afkastestimationer, er af økonomisk værdi, udregnes tilhørende porteføljeafgiter. Dette gøres ifølge fremgangsmåden, beskrevet af Fleming et al. (2001) og Corte et al. (2008). En porteføljeafgift fastsættes således, investoren er indifferent imellem porteføljerne, der sammenlignes, eftersom denne får den samme nytte af porteføljerne. Den kvadratiske nyttefunktion er anvendelig, eftersom den kan anskues som en approksimering af investorens sande nyttefunktion. Investorens nytte, U, til tidspunkt $t+1$, kan approksimeres til at være:

$$U(W_{t+1}) = W_t R_{p,t+1} - \frac{\lambda W_t^2}{2} R_{p,t+1}^2 \tag{3.7}$$

[2]Merton (1972) viser at løsningen en mean-variance investor står overfor kan findes ved opsætning af Lagrangefunktion og løse for førsteordens ligningssættet. Repræsentationen i ligning (3.6) er givet af Baviera & Bianchi (2019).

hvor W_t angiver investorens formue til tidspunkt t, λ er en parameter, der angiver investorens absolutte risikoaversion, hvilken interagerer med størrelsen på formuen. Det er værd at pointere, at λ ikke er den samme som risikoaversitets parameteren γ i ligning (3.6), som er i kontekst af porteføljens varians. For at have sammenlignelige porteføljer sættes den relative risikoaversion, $\delta_t = \lambda W_t/(1 - \lambda W_t)$, lig en konstant. Den gennemsnitlige realiserede nytte, $\bar{U}(\cdot)$, for hver portefølje kan nu findes ved:

$$\bar{U}(\cdot) = W_0 \sum_{t=0}^{T-1} \left\{ R_{p,t+1} - \frac{\delta}{2(1+\delta)} R_{p,t+1}^2 \right\} \tag{3.8}$$

hvor W_0 angiver formuen til tid 0, og $R_{p,t}$ er benchmark porteføljens afkast til tidspunkt t. Ved at sætte den initiale formue $W_0 = 1$ generaliseres udtrykket. Herefter bestemmes porteføljeafgiften på den alternative portefølje, Φ, således at nytten mellem de to porteføljer er ens, som i følgende:

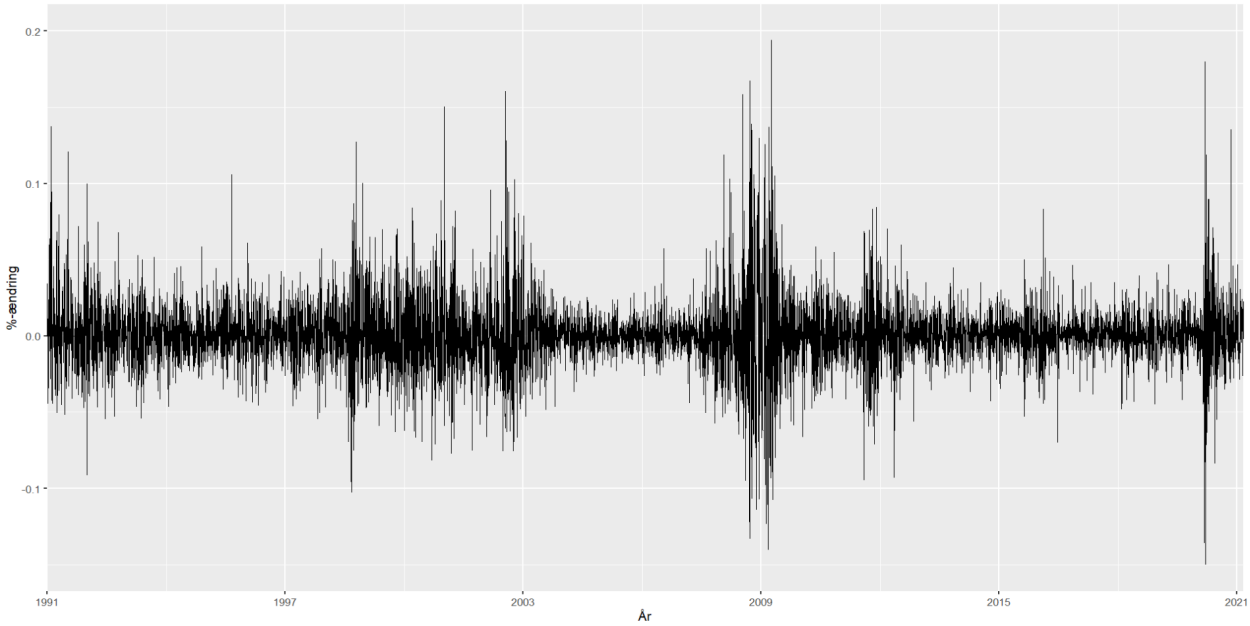
$$\sum_{t=0}^{T-1} \left\{ \left(R_{p,t+1}^* - \Phi\right) - \frac{\delta}{2(1+\delta)} \left(R_{p,t+1}^* - \Phi\right)^2 \right\} = \sum_{t=0}^{T-1} \left\{ R_{p,t+1} - \frac{\delta}{2(1+\delta)} R_{p,t+1}^2 \right\} \tag{3.9}$$

hvor $R_{p,t+1}^*$ angiver afkastet på den alternative portefølje. Afgiften Φ er en andel af den samlede formue investeret og rapporteres i basispoint.

3.3 Volatilitetsmodellering

En fundamental antagelse bag økonometriske modeller er, at fejlleddet udviser homoskedasticitet, hvorved effekter, der ikke forklares af modellen, kan anses som hvid støj. Finansielle tidsserier har tidsvarierende volatilitet, hvilket ofte ses ved, at pludselig volatilitet følges af perioder med høj volatilitet, dvs. volatilitetsklynger. De daglige prisændringer i aktiekursen for JPMorgan Chase & Co. i årene omkring Finanskrisen, figur 3.2, kan ses som et eksempel på dette.

Fig. 3.2: JPM-aktiens daglige %-ændringer 1991-2021



Eftersom banken JPMorgan Chase & Co. var særligt eksponeret overfor Finanskrisen i 2007-2009 og dennes efterskælv, ses der også høj volatilitet i denne periode. Modsat kan der også ses perioder før og efter krisen med relativt lav volatilitet. Tidsserien kan anses for at udvise betinget heteroskedasticitet, når der er perioder med høj volatilitet, men den ubetingede varians, dvs. langsigtede, er konstant. Denne langsigtede varians kan bruges til at sammensætte porteføljer med en køb og hold strategi, hvor de kortsigtede dynamikker er uvigtige. Projektet arbejder med højfrekvens data og ugentlige porteføljeallokeringer, og der fokuseres derfor på de kortsigtede dynamikker, hvorfor der anvendes forecasts af de betingede varianser.

3.3.1 GARCH

Følgende gennemgang af GARCH-modellen er baseret på kapitel 3 af Applied Econometric Timeseries af Walter Enders.

I artiklen "*Generalized Autoregressive Conditional Heteroskedasticity*" fra 1986, introducerede Tim Bollerslev den generaliserede ARCH-model, GARCH. GARCH muliggør modelleringen af ARCH-effekter som en ARMA-proces. Dette kan være fordelagtigt, da ARCH-modeller ofte krævede mange lags i empiriske applikationer, hvor der i stedet kunne anvendes en mere sparsommelig ARMA-repræsentation af den betingede heteroskedasticitet. For en

GARCH(p, q) kan processen for fejlleddet gives ved:

$$\varepsilon_t = v_t\sqrt{h_t} \tag{3.10}$$

hvor

$$h_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-1}^2 + \sum_{i=1}^{q} \beta_i h_{t-i} \tag{3.11}$$

Denne nye repræsentation af den betingede heteroskedasticitet, som ses i ligning (3.11), er yderst relevant for modelleringen af vedvarende højnet volatilitet, som ofte kan findes på de finansielle markeder. Dette gøres gennem størrelsen på β parametrene. Større β parametre vil medføre, at modellen forventer at volatiliteten er mere vedvarende i de følgende perioder. $\alpha_1, \alpha_2, ..., \alpha_p$ parametrene angiver effekten tidligere innovationer har på den betingede varians.

Ligesom ARCH-specifikationen, skal parameterværdierne i ligning 3.11 være positive, således at der ikke estimeres en negativ betinget varians. Ligning (3.10) kan anvendes til at vise, at det ubetingede gennemsnit er:

$$E[\varepsilon_t] = E[v_t]E\left[\sqrt{h_t}\right] = 0 \tag{3.12}$$

Den betingede varians kan vises at være $E[\varepsilon_t^2|\varepsilon_{t-1}] = h_t$, hvormed GARCH-modeller stadig udviser ubetinget homoskedasticitet og betinget heteroskedasticitet. Den ubetingede varians kan for en GARCH(p,q) vises at være:

$$E[\varepsilon_t^2] = \frac{\alpha_0}{1 - \sum_{i=1}^{p} \alpha_1 - \sum_{i=1}^{q} \beta_1} \tag{3.13}$$

Som kan ses i ligning (3.13) vil den ubetingede varians være stabil, når

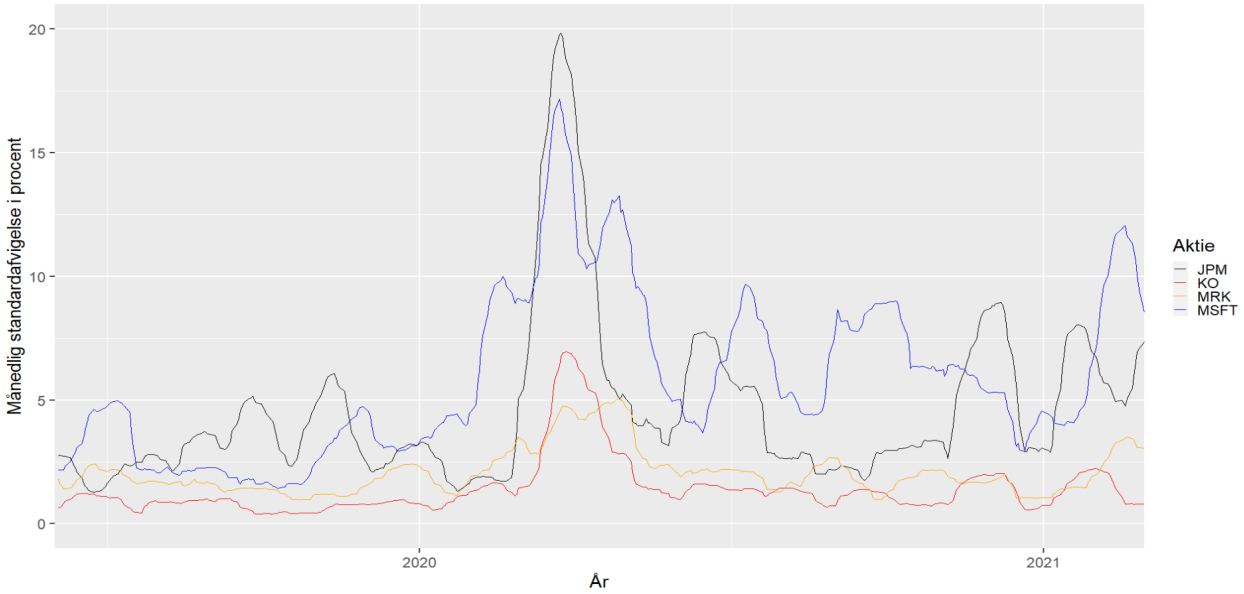
$$1 - \sum_{i=1}^{p} \alpha_1 - \sum_{i=1}^{q} \beta_1 \neq 0 \tag{3.14}$$

3.3.2 DCC-GARCH

Redegørelsen for DCC-GARCH er baseret på Engle (2002), "Multivariate DCC-GARCH Model" af Elisabeth Orskaug fra 2009, samt "The rmgarch models: Background and properties" af Alexios Ghalanos fra 2019.

Ved konstruktionen af en portefølje af aktiver er det centralt at vide, hvor meget hvert aktiv varierer men også hvor meget, aktiverne kovarierer for at kunne minimere den risiko, der påtages. Eftersom varianserne og kovarianserne er tidsvarierende, bør porteføljekonstruktionen baseres på et forecast af disse. I modelleringsprocessen er det også relevant at tage højde for uventet information, som kan påvirke priserne på mange aktier samtidigt. Figur 3.3 viser standardafvigelser udregnet med et 30-dags rullende tidsvindue for fire aktier, der indgår i projektets porteføljer: $JPM, $KO, $MRK og $MSFT i årene 2019-2021.

Fig. 3.3: Standardafvigelser af afkast i årene 2019-2021



De fire aktier tilhører forskellige sektorer; $KO er en forbrugsdefensiv aktie, $MRK tilhører sundhedsindustrien og $MSFT teknologisektoren. Der kan ses perioder, særligt tydeligt omkring februar 2020, hvor alle aktiernes volatiliteter pludseligt forøges efter usikkerheden forårsaget af COVID-19 pandemien. Der er også generelt højere standardafvigelser i den resterende periode end før 2021, hvilket meget vel kan fortsætte, eftersom alle effekter af COVID-19-krisen ikke er kendte endnu.

For at modellere mere end diagonalen af aktiers varians-kovariansmatrice, kan en multivariat GARCH-model anvendes, hvor der både estimeres de betingede varianser og de betingede kovarianser. Hertil introducerede Bollerslev (1990) Constant Conditional Correlation (CCC) modellen. Som navnet angiver, antages det i CCC-modellen, at de betingede korrelationer er konstante over tid for således at kunne forudsige kovarianserne. Robert Engle introducerede en generalisering af modellen i artiklen *Dynamic Conditional Correlation - A Simple Class of Multivariate GARCH Models* fra 2002. I modellen, kendt som DCC-GARCH, ændres de betingede korrelationer til at være tidsvarierende, hvilket er mere empirisk relevant. Mange

andre multivariate GARCH-modeller har også dimensionalitetproblemer, hvilket løses ved, at DCC-modellen estimeres over to trin.
DCC-modellen er en multivariat GARCH-model, der tilhører klassen af modeller med betingede varianser og korrelationer. Den betingede varians-kovariansmatrice kan defineres til at være:

$$E[r_t r_t^T|_{t-1}] \equiv H_t \tag{3.15}$$

hvor r_t er en vektor bestående af n aktiers tidsserier, og H_t er en positiv definit varians-kovarians matrice. I modellen nedbrydes varians-kovariansmatricen til to tidsvariende matricer for at adskille de univariate og multivariate dynamikker, således H_t gives ved:

$$H_t = D_t R_t D_t \tag{3.16}$$

D_t er en diagonal matrice, der består af de tidsvarierende betingede standardafvigelser fra n univariate GARCH-modeller, som vist i ligning 3.11. D_t kan derfor repræsenteres ved:

$$D_t = \begin{bmatrix} \sqrt{h_{1,t}} & 0 & 0 & \cdots & 0 \\ 0 & \sqrt{h_{2,t}} & 0 & \cdots & 0 \\ 0 & 0 & \sqrt{h_{3,t}} & & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sqrt{h_{n,t}} \end{bmatrix} \tag{3.17}$$

Det kræves, at alle GARCH-processernes fejlled er normalfordelte samt, at modellerne overholder de nævnte parameterrestriktioner og stabilitetsbetingelser[3]. R_t er en korrelationsmatrice bestående af de betingede korrelationer imellem de standardiserede fejl. R_t begrænses ved, at de betingede varianser af fejl skal være lig 1, hvormed matricen bliver:

$$R_t = \begin{bmatrix} 1 & \rho_{12,t} & \rho_{13,t} & \cdots & \rho_{1n,t} \\ \rho_{21,t} & 1 & \rho_{23,t} & \cdots & \rho_{2n,t} \\ \rho_{31,t} & \rho_{32,t} & 1 & & \rho_{3q,t} \\ \vdots & \vdots & & \ddots & \vdots \\ \rho_{n1,t} & \rho_{n2,t} & \rho_{n3,t} & \cdots & 1 \end{bmatrix} \tag{3.18}$$

R er den eneste forskel fra Bollerslevs CCC-model, eftersom denne tillades at variere over tid i DCC-modellen. For at H_t overholder kravet om at være en positiv definit matrice, skal R_t også være positiv definit til enhver tidsperiode t, eftersom de betingede standardafvigelser i

[3]Det er ikke et krav, at de univariate GARCH-modeller skal have samme specifikationer.

D_t ikke kan tage negative værdier. Dette opnås ved at modellere proxy processen, Q_t, der kan gives ved:

$$Q_t = (1 - a - b)\bar{Q} + a\epsilon_{t-1}\epsilon_{t-1}^T + bQ_{t-1} \tag{3.19}$$

hvor ϵ_t er en vektor bestående af de standardiserede fejlled, a og b er ikke-negative skalarer, og $\bar{Q}$ er en matrice bestående af de ubetingede korrelationer mellem de standardiserede fejl, $E[\epsilon_t\epsilon_t^T]$. Sålænge $a+b < 1$ er Q_t stationær og positiv definit. Udover dette, hvis startværdien af Q_t, Q_0, er positiv definit, sikres det, at H_t er positiv definit. Signifikante estimater af a og b implicerer dermed, at korrelationsmatricen er tidsvarierende. Insignifikante estimater af a og b, ville derfor også være ensbetydende med at det er upassende at anvende en DCC-model. Korrelationsmatricen, R_t, nedbrydes til:

$$R_t = diag\{Q_t\}^{-1}Q_t\, diag\{Q_t\}^{-1} \tag{3.20}$$

hvor diagonalen af Q_t matricen anvendes for at sikre at alle elementerne i R_t er mindre eller lig 1.

3.3.3 Estimation og forecasting

Eftersom projektet afgrænses til, at de standardiserede fejl følger en fælles normalfordeling, således at der ikke skal tages højde for skævhed eller kurtosis parametre, er det ikke nødvendigt at estimere alle parametre i DCC-GARCH-modellen simultant. Log-likelihood funktionen kan gives som summen af et volatilitets-led, ℓ_V, og et korrelations-led, ℓ_K:

$$\ell(\theta, \phi) = \ell_V(\theta) + \ell_K(\theta, \phi) \tag{3.21}$$

hvor θ henviser til parametrene i D matricen, og ϕ er parametrene fra R matricen fra ligning (3.16). Log-likelihoodfunktionen $\ell(\theta, \phi)$ maksimeres ved brug af en to-trins fremgangsmåde. Først estimeres parametrene, der maksimerer $\ell_V(\theta)$. Volatilitets log-likelihood funktionen kan gives ved[4]:

$$\ell_V(\theta) = -\frac{1}{2}\sum_t\sum_{i=1}^{n}\left(log(2\pi) + log(h_{i,t}) + \frac{r_{i,t}^2}{h_{i,t}}\right) \tag{3.22}$$

Ligning (3.22) er summen af de univariate GARCH-modellers likelihood-funktioner. $\ell_V(\theta)$ maksimeres derfor ved, at hver enkel univariat GARCH-model maksimeres separat.

[4]En detaljeret udledning af log-likelihood funktionen for DCC-GARCH kan denne findes i Engle (s. 10-12, 2002)

Andet trin er at estimere korrelationsparametrene ved at anvende log-likelihood funktionen for korrelationsleddet:

$$\ell_K(\theta, \phi) = \frac{1}{2} \sum_t (log|R_t| + \epsilon_t^T R_t^{-1} \epsilon_t - \epsilon_t^T \epsilon_t) \tag{3.23}$$

Her anvendes estimaterne af θ, der maksimerede ℓ_V, således det kun er korrelationsparametrene ϕ, der skal bruges til at maksimere log-likelihood funktionen i ligning (3.23). Parametrene fra andet trin findes som en funktion af parametrene fra første trin, og estimaterne vil derfor være konsistente, såfremt at estimaterne fra første trin er konsistente.
Forecasting med DCC-GARCH kan gøres på to måder, hvoraf *rmgarch* funktionerne, projektet benytter sig af, anvender den, der skulle have mindst bias (Ghalanos, 2019). Proxy processen Q, n perioder frem, kan gives ved:

$$Q_{t+n} = (1 - a - b)\bar{Q} + aE\left[\epsilon_{t+n-1}\epsilon_{t+n-1}^T | t\right] + bQ_{t+n-1} \tag{3.24}$$

hvor $E\left[\epsilon_{t+n-1}\epsilon_{t+n-1}^T | t\right] = R_{t+n-1}$ og $R_{t+n} = diag\{Q_{t+n}\}^{-1} Q_{t+n}\, diag\{Q_{t+n}\}^{-1}$. For at finde R_{t+n} sættes $\bar{Q} \approx \bar{R}$ og $E[Q_{t+1}|_{t+1}] = E[R_{t+1}|_{t+1}]$, således et n perioder frem forecast af R_t kan gives ved:

$$E[R_{t+n}|_t] = \sum_{i=0}^{n-2} (1 - a - b)\bar{R}(a + b)^i + (a + b)^{n-1} R_{n+1} \tag{3.25}$$

Ligning 3.25 viser, at estimatet af R_{t+n} forfalder med raten $(a + b)$, hvor stationaritetsbetingelsen sikrer, at $a + b < 1$. Eftersom projektet udelukkende forecaster én uge frem, er dette ikke et problem.

3.3.4 Ljung-Box Test

Projektet benytter sig af Ljung-Box test statistikker til at evaluere, om de univariate volatilitetsmodeller har indfanget ARCH-effekterne i data, således at det kan antages, at de standardiserede fejl opfører sig som hvid støj.
Ljung-Box testen kan identificere ARCH-effekter ved at teste for autokorrelation i de kvadrerede estimerede residualer, $\{\hat{\varepsilon}_t^2\}$. For T observationer, y_1 til y_T, og hvor r_s er estimater af residualernes autokorrelationer, kan Q-statistikken udregnes ved:

$$Q = T(T + 2) \sum_{k=1}^{s} \frac{r_k^2}{(T - k)} \tag{3.26}$$

Hvis værdien af Q-statistikken, hvilken følger en χ^2-fordeling med s frihedsgrader, er mere ekstrem end den kritiske værdi, er mindst én autokorrelation statistisk anderledes fra 0 ved det valgte signifikansniveau.

Ljung-Box testen kan både anvendes til at undersøge, om det er nødvendigt at anvende en (G)ARCH-specifikation, og om alle ARCH-effekter er forklaret af (G)ARCH-modellen, således residualerne udelukkende opfører sig som hvid støj.

3.4 Machine Learning

Tom M. Mitchell (1997) definerer machine learning som *"(...) the study of computer algorithms that allow computer programs to automatically improve through experience"*. Termet er bredt og indeholder som disciplin mange metoder, hvoraf kunstige neurale netværk blot er et segment. Det overordnede formål med machine learning teknikker er at automatisere analyse af høj-dimensionelt og kompleks data, som normalt kun det menneskelige sind har været i stand til at tolke på. Men nyere metoder har vist, at det også er muligt for en computer at finde mønstre og sammenhænge i data, som er usynlige eller for komplekse for den menneskelige hjerne. Som Chong et al. (2017) argumenterer, er disse machine learning teknikker særligt egnet til aktiemarkedsforudsiglse, fordi mange faktorer anses for at påvirke aktiekurser og ofte i komplekse og ikke-lineære sammenhænge. De metoder fra machine learning, der vil blive anvendt i dette projekt, går ind under emneområdet kendt som deep learning, der anvender kunstige neurale netværk til at fange sammenhænge og lære af store mængder data.

3.4.1 Artificial Neural Network

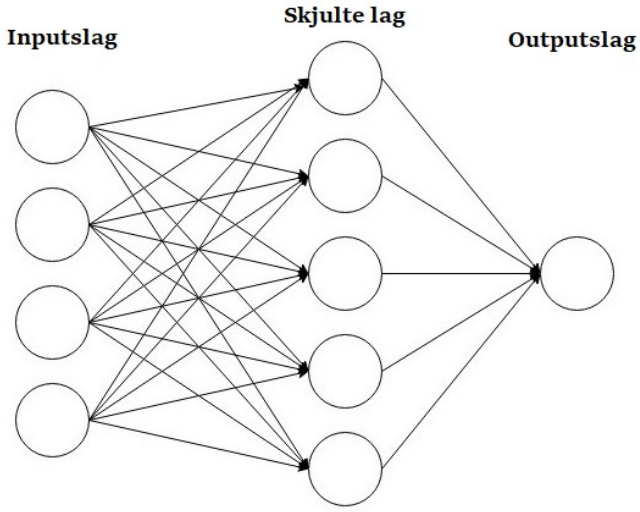
Følgende sektion vedrørende kunstige neurale netværk er hovedsageligt baseret på kapitel 6 af bogen Deep Learning (2016), der er skrevet af Ian Goodfellow, Yoshua Bengio and Aaron Courville. Gennemgangen vil være relativt overfladisk og ønskes en mere dybdegående gennemgang, så henvises læser direkte til kildematerialet.

Inden for deep learning er der en overordnet gren af estimationsmodeller, som klassificeres som artificial neural network[5] (ANN). Disse modeller er konstruerede på en måde, der forsøger at imitere den måde, hvorpå biologiske neurale netværk fungerer. Ligesom biologiske netværk, består ANNs af neuroner, som modtager og bearbejder information. Neuronerne i et netværk er forbundet til hinanden via edges, og det er igennem disse forbindelser, at information bliver ført igennem et netværk. Edges mellem noder, repræsenterer synapsen mellem to celler. Hver gang information når en neuron, bliver informationen bearbejdet og efterfølgende videresendt til andre neuroner.

[5] Også kaldet kunstige neurale netværk.

Neurale netværk er bygget op i lag, hvori neuronerne er placerede. Disse lag repræsenterer aksoner, som er grupperinger af celler. Det første lag i et netværk er inputslaget, hvor inputsdimensionerne specificeres, så de forklarende variable kan "fodres" direkte til netværket. Alle neuroner i inputslaget er forbundet til hver neuron i det efterfølgende skjulte lag. Hver en edge i netværket er knyttet med en bestemt vægt. Neuronerne i de fleste feed-forward netværk indeholder et bias, som er afgørende for det output, som en neuron videresender. Disse vægte og bias vil igennem træning af netværket blive justerede, sådan at netværket har den største nøjagtighed. I de skjulte lag bliver inputtet fra det foregående lag forarbejdet, ikke-lineært transformeret og videresendt til det efterfølgende lag. Der kan være en arbitrær mængde af neuroner og skjulte lag i et netværk, der bestemmes af modelkonstruktørerne, men jo flere neuroner og lag i netværket, desto større kompleksitet. I outputslaget aggregeres forbindelserne til en endelig prædiktion. Neuronerne i de skjulte lag transformerer inputtene fra foregående lag ved at anvende en aktiveringsfunktion, og det er denne funktion, der afgør, hvorledes inputtene fra foregående lag videresendes til de efterfølgende lag.
Figur 3.4 illustrerer ovenstående netværksbeskrivelse i et netværk, hvor information kun kan "passere" fremad, også kendt som et feed-forward ANN. Et feed-forward netværk er den mest basale model indenfor kunstige neurale netværk. I illustrationen præsenteres et netværk med et enkelt skjult lag, hvor inputslaget og det skjulte lag har henholdsvis fire og fem neuroner. Det er endvidere gjort klart, hvordan neuronerne i de forskellige lag er forbundet til hinanden samt forbindelsernes retning. Endelig afsluttes netværket med et outputslag, som består af en enkelt neuron.

Fig. 3.4: Simple Feed-Forward ANN



En speciel egenskab ved et neuralt netværk er, at det er i stand til automatisk at lære fra inputdataet. Dette sker ved, at netværket gennem træning ændrer på dets parametre, sådan at dens prædiktioner har den største præcision. For feed-forward netværk sker denne træningsproces gennem anvendelse af *backpropagation*, som vil blive beskrevet i senere afsnit. Til hver forbindelse mellem neuronerne tilhøres en vægt. For eksempel har det illustrerede netværk i figur 3.4 i alt 25 vægte ($4 \times 5 + 5 \times 1$) samt 6 bias, hvilket vil sige, at netværket i alt har 31 parametre, der kan kalibreres under træning for at optimere dens nøjagtighed.

Aktiveringsfunktion

En aktiveringsfunktion er som nævnt i foregående afsnit en funktion, som transformerer information. I et neuralt netværk anvendes aktiveringsfunktioner i neuronerne i de skjulte lag. Der findes mange forskellige aktiveringsfunktioner, der på hver sin måde transformerer information, hvor størstedelen af disse funktioner ikke-lineært transformerer inputtet. Den bestemte funktion, der anvendes i et netværk, afhænger blandt andet af det pågældende problem.
For at præsentere, hvordan en neurons aktiveringsfunktion fungerer, gennemgås en meget simpel aktiveringsfunktion som vist af Abraham (2005). I et givet netværk med n input, $x_1, ..., x_n$, kan outputtet, O, af en neuron i det skjulte lag opsættes som:

$$O = f(z) = f\left(\sum_{j=1}^{n}(w_j x_j) + b\right) \tag{3.27}$$

hvor w_j er de tilsvarende n vægte, $f(z)$ er aktiveringsfunktionen, der anvendes på variablen z, som er skalarproduktet af vægtene og inputtene samt et eventuelt bias b. I det meget simple tilfælde findes outputtet O ved:

$$O = f(z) = \begin{bmatrix} 1 & if \quad z \geq \theta \\ 0 & otherwise \end{bmatrix} \tag{3.28}$$

hvor θ er en grænseværdi. Ovenstående aktiveringsfunktion er en grænsebaseret aktiveringsfunktion og kaldes en binær step funktion. Ved anvendelse af en sådan aktiveringsfunktion kan netværket ikke lære ikke-linære sammenhænge. Yderligere implicerer en binær step funktion, at små ændringer af vægte eller bias kan medføre, at en neuron skifter fra ikke at aktivere til at aktivere, hvilket kan resultere i en drastisk ændring af den endelige forudsigelse. En lineær aktiveringsfunktion vil modsat den binære aktiveringsfunktionen transformere inputtet lineært. Herved vil outputtet af en neuron, ved anvendelse af en

lineær aktiveringsfunktion, være proportionel med inputtet. En sådan aktiveringsfunktion kaldes også en identitetsfunktion og kan fremsættes som:

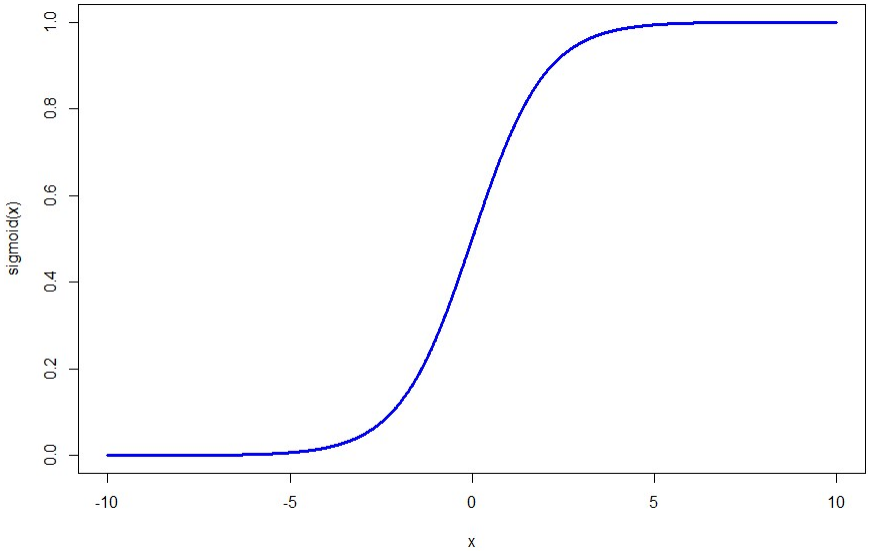
$$O = z = \sum_{j=1}^{n}(w_j x_j) + b \tag{3.29}$$

hvor z er den vægtede sum af input samt et eventuelt bias. Denne type af aktiveringsfunktion medfører dog, at netværket ikke optimalt kan anvende backpropagation til at træne. Yderligere så kan netværket heller ikke lære ikke-lineære sammenhænge, fordi netværket bliver en lineær kombination af lineære lag.
Meget af det data, hvor neurale netværk er anvendelige, er yderst kompleks og består ofte af flere dimensioner. Derfor giver det ikke mening at forsøge at opsætte netværk baseret på lineære aktiveringsfunktioner, hvis sammenhænge i dataet er ikke-lineære. Traditionelt set har Sigmoid-aktiveringsfunktionen været den mest anvendte aktiveringsfunktion, når man har ønsket en ikke-lineær transformation af dataet for at netværket kan lære ikke-lineære sammenhænge. En generel logistisk Sigmoid-funktion præsenteres som:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.30}$$

hvor x er et givet input. Denne transformation har nogle interessante egenskaber. Figur (3.5) viser distributionen af en Sigmoid-funktion. Som det kan ses, tager outputtet udelukkende en værdi mellem 0 til 1, lige meget størrelsesordenen på inputværdien.

Fig. 3.5: Distribution af logistisk Sigmoid-funktion



I et neuralt netværk vil inputtet i en Sigmoid-aktiveringsfunktion bestå af en variabel z. Herved bestemmes outputtet af en neuron, som anvender en Sigmoid-funktion, af to trin.

Først beregnes z fra inputtene fra forrige lag x og tilhørende vægte w, hvorefter anvendelsen af Sigmoid-aktiveringsfunktionen ikke-lineært transformerer z som outputtet O. Altså:

$$O = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.31}$$

$$z = \sum_{j=1}^{n} (w_j x_j) + b \tag{3.32}$$

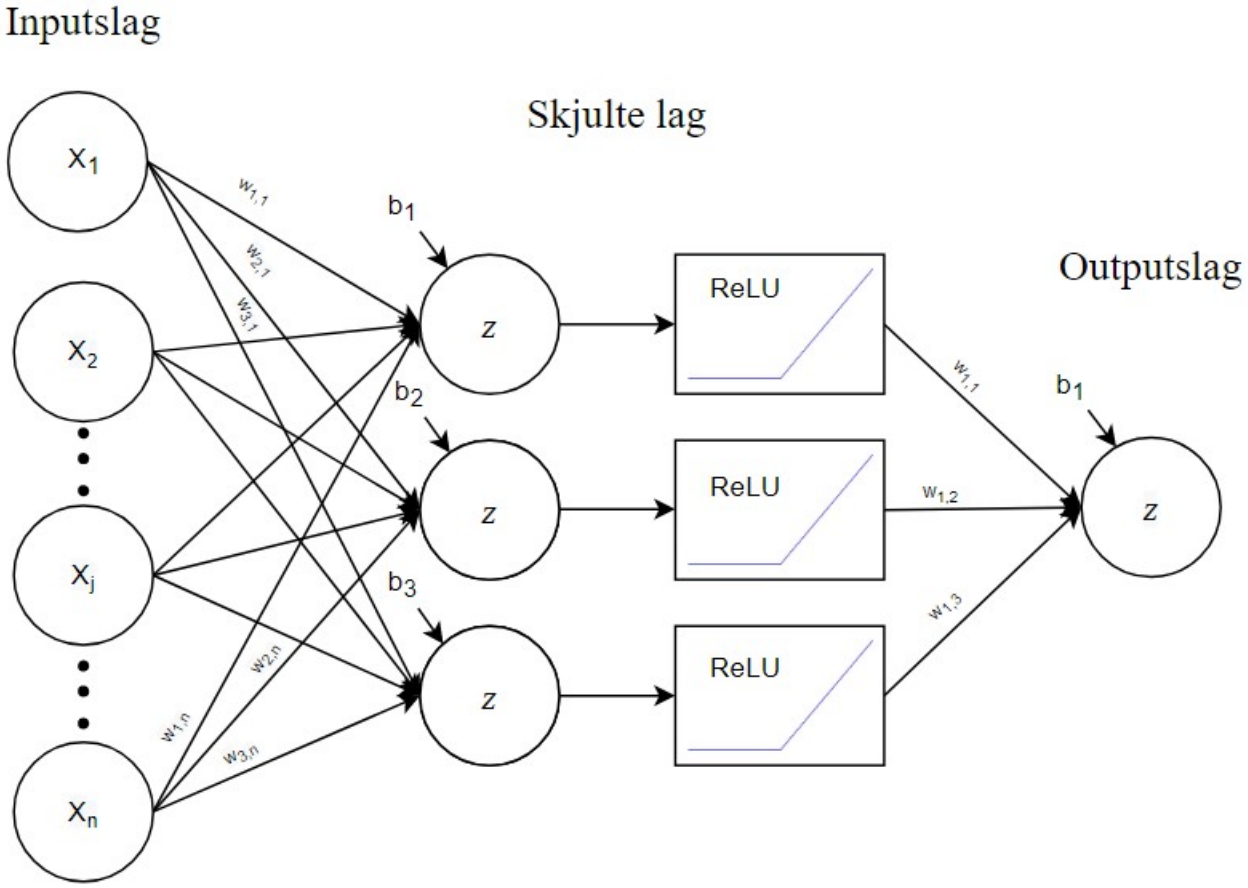
Dermed er input, vægte og et eventuelt bias med til at afgøre outputtet af en neuron, som sendes videre i netværket. Ved anvendelsen af en Sigmoid-aktiveringsfunktion vil en lille ændring af vægte eller bias ikke have samme drastiske ændringer af outputtet som ved en binær step aktiveringsfunktion. Sigmoid-funktionen anvendes desuden også i flere porte i neuronerne i et LSTM-netværk[6]. Ovenstående proces er generel for alle skjulte neuroner i et feed-forward netværk - først udregnes den vægtede sum af input og et eventuelt bias, herefter transformeres informationen ift. den anvendte aktiveringsfunktion.
I de seneste år er Rectified Linear Unit (ReLU) blevet den standarde aktiveringsfunktion, når det kommer til komplekse problemstillinger. Ligesom de andre transformationsfunktioner i neurale netværk, transformeres den vægtede sum af input og bias z som i (3.32). Den præsenteres som følgende:

$$O = f(z) = \begin{bmatrix} 0 & if \quad z < 0 \\ z & otherwise \end{bmatrix} = max(0, z) \tag{3.33}$$

I ReLU-funktionen vil negative z-værdier blive transformerede til 0, og positive bliver transformerede lineært. ReLU-funktionen er attraktiv, fordi den i høj grad minder om en lineær aktiveringsfunktion, hvilket medfører, at mange af de gode egenskaber fra lineære aktiveringsfunktioner kan anvendes. Dette indebærer blandt andet, at netværket er hurtigere til at optimere ved anvendelse af gradient-descent baseret algoritmer, hvilket betyder, at netværket er hurtigere til at lære. Selvom de lineære egenskaber kan anvendes under ReLU-funktionen, er det stadig en ikke-lineær transformation af inputtet, fordi alle negative værdier sættes lig 0.
Figur (3.6) opsummerer, hvordan en neuron i et skjult lag fungerer. Først udregnes neuronens inputvariabel z, som den vægtede sum af inputtene x_j plus bias, som herefter transformeres ift. den anvendte aktiveringsfunktion. Her benyttes en ReLU funktion, og på figuren kan den grafiske repræsentation af funktionen ses, hvor negative z-værdier transformeres til 0, og positive værdier transformeres lineært.

[6] Dette vil blive præsenteret i afsnit 3.4.2.

Fig. 3.6: Feed-forward neuralt netværk



I outputslaget er der placeret en enkelt neuron, og det er her netværket kommer med den endelige prædiktion. Denne neuron modtager ligeledes information fra de foregående lag, beregner summen af de vægtede input, tilføjer bias og herefter transformerer den z til en endelig prædiktion. Generelt anvendes der enten en lineær eller binær aktiveringsfunktion i outputsneuronen, hvis valg afhænger af det pågældende regressionsproblem.

Trænings- og testsdata

Et vigtigt element i opsætningen af machine learning modeller, er en initial opdeling af det tilgængelige datasæt. Hertil bliver datasættet opdelt i to - et trænings- og et test-datasæt. Træningsdataet anvendes til træningen af netværket, mens testsættet anvendes til at evaluere, hvordan modellen præsterer på nyt data, som modellen ikke er blevet præsenteret for under træning af netværket. Dette gøres for at emulere netværkets præstation på helt ny data. Under træningen kan træningssættet yderligere blive splittet op i et trænings- og valideringssæt, hvor valideringssættet anvendes til at mindske *overfitting* af netværkets parametre. Overfitting af et netværk betyder, at netværkets parametre tilpasses for voldsomt til træningsdataets unikke mønstre og derved vil have en høj performance på træningsættet.

Hvis ny data ikke har samme mønstre som træningssættet, kan det medføre, at netværket vil forudsige dårligt på ny data. Ved at evaluere netværket på et valideringssæt under træningen, fremmes en optimal fintuning af netværkets parametre, uden at anvende testsættet. Et netværks parametre bliver altså ikke fintunet med hensyn til performance på test data, da dette skal anvendes til at repræsentere fremtidig data.

Træning af netværket

Neurale netværk skal, som nævnt tidligere, igennem en træningsprocess, hvor netværkets parametre trænes, så netværket som helhed har den største nøjagtighed. Alle vægte i netværket initialiseres til lave og tilfældige værdier. Herefter anvender netværket træningsdatasættet og de tilfældigt satte parametre til at prædiktere output, som sammenlignes med faktiske output fra træningsdatasættet for at evaluere modellen. Efterfølgende ændres vægtene baseret på evalueringen og en ny iteration igangsættes for at se, om fejlestimaterne er blevet mindre. En loss-funktion anvendes til at evaluere modellens nøjagtighed, hvor lavere tab betyder større nøjagtighed. Loss-funktionen er en funktion, der kortlægger, hvordan fejlestimaterne af forudsigelserne ændres ved ændringer af vægtene. Valget af loss-funktion afhænger af, om det er et klassifikations- eller regressionsproblem, hvor man henholdsvis vil anvende en cross-entropy- eller mean squared error loss-funktion[7]. Ovenstående er en meget forsimplet udgave af, hvordan et kunstigt neuralt netværk lærer. Træningen foregår ved at anvende et sammenspil af to algoritmiske tilgange - stokastisk gradient descent og back-propagation.

Stokastisk Gradient Descent og Back-propagation

Gradient descent er en optimeringsalgoritme, som anvendes til at finde et bestemt sæt af inputsvariable, der medfører, at outputtet af en bestemt funktion er dets minimum. Hertil skal gradienten af den bestemte funktions variable udregnes. Gradienten bruges til at vise, hvilken retning inputsvariablene bør ændres for at nærme den bestemte funktions minimum[8]. Efter første beregning af hver inputsvariabels gradient, ændres de marginalt for så igen at udregne deres gradient. Denne proces gentages indtil den bestemte funktion har nået et minimum, hvorved inputsvariablene vil være ændrede til et bestemt sæt, der medfører et minimum for den bestemte funktion. Loss-funktionen er den bestemte funktion, hvis minimum skal findes, når det drejer sig om neurale netværk. Hermed bliver vægtene og

[7]Der findes mange forskellige loss-funktioner, dog er disse to de mest anvendte inden for de to problemstillinger.

[8]Gradienten viser dog, hvilken retning inputsvariablene skal ændres for at øge outputtet af den bestemte funktion, derfor skal inputsvariablene ændres i den modsatte retning for at nå et minimum.

bias i netværket de inputsvariable, som gradient descent skal finde et sæt af for at minimere loss-funktionen. Til at udregne gradienten for inputsvariablene (vægte og bias) anvendes back-propagation algoritmen. Alt i alt er det et sammenspil mellem de to algoritmer, som tilsammen udgør træningen af netværket. Modellens parametre ændres en lille andel efter hver iteration for at minimere tabet og øge modellens nøjagtighed. Denne andel, som vægtene og bias ændres med, bestemmes af modelkonstruktørerne og kaldes for læringsraten.

3.4.2 Recurrent Neural Network

Følgende afsnit omkring recurrent neurale netværk er baseret på kapitel 10 i bogen Deep Learning af Goodfellow et al. (2016). Recurrent neurale netværk præsenteres i generel form.

Mange praktiske applikationer af deep learning modeller vedrører sig forudsigelsesproblemer baseret på tidsserie data, der i sig selv er sekventielt data. Sekventielt data kan betegnes som data, hvor rækkefølgen, som dataet bliver præsenteret på, har en stor betydning. Et klassisk eksempel på sekventielt data er tekstdata. I en vilkårlig sætning har ordenes placering i sætningen stor indflydelse på deres betydning. Derudover afhænger betydningen af ord også af, hvilke andre ord, der er i sætningen. Mennesker er netop i stand til at bearbejde sætninger ved ikke blot at se på de enkelte ord uafhængigt af hinanden, men i stedet se sætningen som helhed, hvor et ords betydning afhænger af de andre ord i sekvensen samt placeringen af ordet. En tidsserie er ligeledes en form for sekventielt data, hvor rækkefølgen af tidsserien har betydning og bestemmes kronologisk.
Recurrent neurale netværk (RNN) er en gren af kunstige neurale netværk, som forsøger at inkorporere den måde, hvorpå den menneskelige hjerne bearbejder sekventielt data. Disse netværk er essentielt dynamiske systemer, hvor nutidige stadier afhænger af forrige stadier. Recurrent neurale netværk har et internt stadie, der kan "huske" information fra forrige input. For at belyse, hvordan RNNs fungerer, præsenteres en klassisk form af et dynamisk system:

$$s_t = f(s_{t-1}; \theta) \tag{3.34}$$

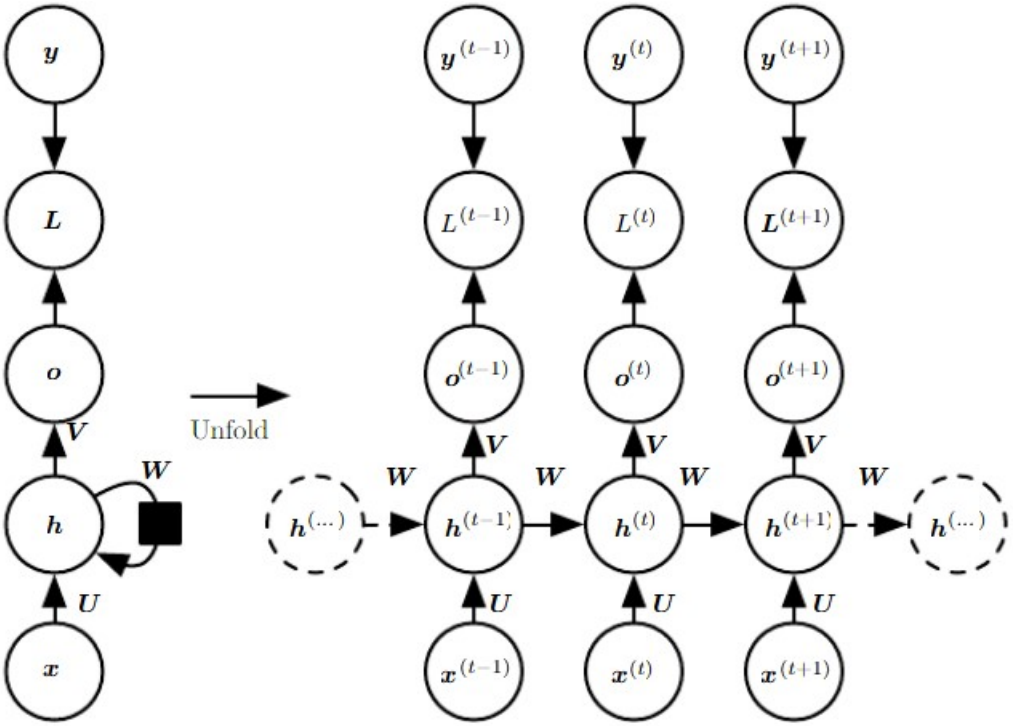
hvor s_t er en tilstandsvariabel på tidspunkt t, og θ er et sæt af parametre. Her ses det, at s ved tid t er en funktion af sin forrige tilstand ved tid $t-1$ samt parametersættet θ. Ligning (3.34) er tilbagevendende eller *recurrent*, da den nuværende definition af s_t refererer tilbage til den forrige definition ved s_{t-1}. I et Recurrent neuralt netværk vil værdien af en node i de skjulte lag blive fundet ud fra et sådan dynamisk system. Ligning (3.35) er en generel præsentation af værdien af en skjult neuron i et recurrent neuralt netværk, hvor den

nuværende tilstand afhænger af både den forrige tilstand samt et eksternt input:

$$h_t = f(h_{t-1}, x_t; \theta) \tag{3.35}$$

hvor h er tilstanden, og x er et ekstern input. I figur (3.7) præsenteres et generelt RNN.

Fig. 3.7: RNN med output



Kilde: Goodfellow et al. (2016).

Den venstre del af figuren viser netværket som et kredsløbsdiagram. Modellen bliver fodret en inputssekvens med x-værdier, hvorfra de skjulte enheder h opdateres, hvor h bliver ved med at udvikle sig og påvirke fremtidige stadier af h. Denne tilbagevendelse er vist med den sorte boks ved tilstanden h. Baseret på information fra de skjulte enheder kommer netværket med en tilsvarende outputssekvens o. Ved træning af netværket findes tabet L ved forskellen mellem de prædikterede værdier $\hat{y}$ og træningsmålet y. Bemærk, at der er tre sæt af vægte, der er med til at paramatisere forbindelserne. Vægtmatricen U parametriserer forbindelsen fra input til skjulte enheder, forbindelsen mellem skjult-til-skjult er parametriseret af vægtene W og fra skjult til output af V i samme stil, som beskrevet i afsnit 3.4.1. Den højre del af figur 3.7 viser essentielt det samme, kredsløbet er blot blevet foldet ud, så det ligner et neuralt netværk. Ligning (3.35) er nu klart repræsenteret i netværket, hvor hver skjult tilstand afhænger af den forrige tilstand og den nuværende værdi af inputssekvensen. Hver node repræsenterer nu et bestemt tidsskridt i tidsserien.

I et RNN anvender neuroner i de skjulte lag og i outsputslaget ligeledes aktiveringsfunktioner til at transformere information. Hvis der tages udgangspunkt i tid t, og der anvendes en hyperbolsk tangens aktiveringsfunktion (tanh), kan outputtet af en skjult neuron findes ved:

$$h_t = tanh(a_t) \tag{3.36}$$

$$a_t = b + Wh_{t-1} + Ux_t \tag{3.37}$$

hvor b er bias og W og U er de tilhørende vægtmatricer. Funktionerne minder i høj grad om processen i ligning (3.32). Først udregnes a_t , som er en funktion af et bias b og de parametriserede værdier af det forrige skjulte stadie h_{t-1} og værdien i inputssekvensen x ved tid t. Efterfølgende transformeres a ved at anvende en aktiveringsfunktion. Værdien i outputssekvensen o ved tid t findes ved:

$$o_t = c + Vh_t \tag{3.38}$$

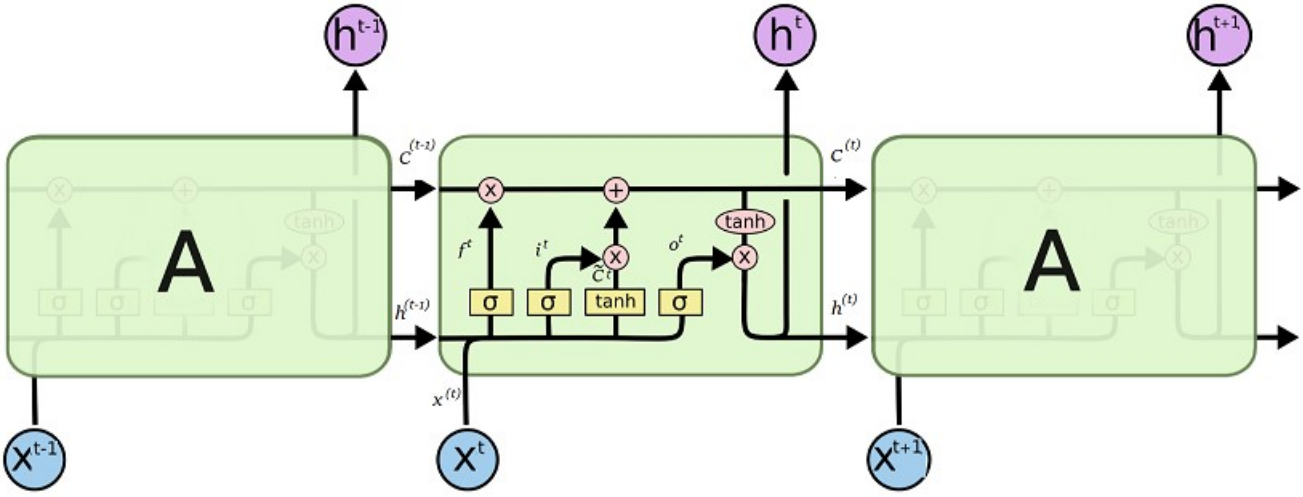
hvor c er et bias, og V er den tilhørende vægtmatrice. Værdien o_t vil herefter blive transformeret til en endelig forudsigelse, $\hat{y}_t$, ved at benytte en given aktiveringsfunktion. Netværket har nu egenskaben af, at tidligere observerede stadier igen og igen vil påvirke fremtidige stadier, dog vil et stadie fra $t-2$ have en indirekte effekt, da h_{t-2} har en direkte effekt på h_{t-1} , som herefter vil påvirke h_t. Ovenstående RNN er en "sekvens-til-sekvens" opsætning. Det vil sige, at netværket får en sekvens som input, hvor outputtet tilsvarende vil være en sekvens. Ved tidsserie data anvendes ofte en "sekvens-til-én" opsætning, hvilket betyder, at netværket anvender hele inputssekvensen til at lave én forudsigelse for den efterfølgende værdi i inputssekvensen.
I praksis har det vist sig, at RNN ikke præsterer specielt godt, når det sekventielle data bliver længere end 5-10 input. Dette skyldes problemer med træningen af netværket. Problemet er, at hvis backpropagation under træningen sker over mange stadier, har gradienterne en tendens til at forsvine til meget små værdier eller eksplodere i den modsatte retning. Hermed vil netværket ikke være i stand til korrekt at ændre på dets vægte gennem træning, hvilket vil medføre, at loss funktionen ikke kan minimeres. Dette problem er kendt som *vanishing* eller *exploding* gradient problemet, og det er et generelt problem, som komplekse og dybe deep learning modeller kan støde på. Recurrent neurale netværk støder relativt hurtigt ind i dette problem, da netværkene hurtigt bliver meget store ved længere sekvenser. Herved kan netværkene ikke lære langsigtede afhængigheder.

Long Short-Term Memory

Følgende gennemgang af LSTM tager udgangspunkt i Chris Olahs artikel fra august 2015: Understanding LSTM Networks.

Til at bekæmpe problemet med kort hukommelse kan en Long Short-Term Memory (LSTM) struktur anvendes. En LSTM-model er blot en gren af RNN, der er i stand til at lære langsigtede afhængigheder, og dermed kan anvendes til sekvenser bestående af hundredevis af tidsskridt (Gers et. al., 2000). LSTM-netværk er mere komplekse end RNNs i den forstand, at de, alt andet lige, har flere lag. Ved en LSTM-struktur udvikles de skjulte neuroner ved, at der internt i neuronen tilføjes tre ekstra lag. Derudover tilføjes der et cellestadie, som essentielt kan ses som en neurons interne hukommelse. Cellestadiet er den tilføjelse, der medgør at LSTM kan huske mere langsigtet. De tre ekstra lag kaldes for porte, og de benyttes til at frasortere ligegyldig information fra cellestadiet ved at "glemme" information fra tidligere stadier, der ikke længere er relevant. Portene kan ligeledes tilføje ny information til cellestadiet - hukommelsen. Figur (3.8) viser en udfoldet version af et LSTM-netværk:

Fig. 3.8: En skjult neuron i et LSTM-netværk



Kilde: Olah (2015).

De gule bokse indikerer et lag placeret i den skjulte neuron, hvor informationen bliver transformeret af en aktiveringsfunktion. Pilene indikerer retningen, som informationen passerer igennem neuronens porte. Den øverste linje, der lineært løber igennem netværket, er cellestadiet. De tre porte (ekstra interne lag) er kendt som "forget"-porten, "input"-porten og "output"-porten, hvor alle tre anvender en sigmoidsaktiveringsfunktion. Som nævnt i afsnit (3.4.1) transformerer Sigmoid-funktionen dataet til en værdi mellem 0 og 1, hvor 0 betyder, at informationen ikke passerer videre og 1, at al information passerer videre. Går

to linjer sammen til en, betyder det, at informationen bliver sammenkædet. Deles en linje i to, bliver informationen kopieret og sendt i flere retninger. Hvis man læser netværket fra venstre, er "forget"-porten det første lag og trin. Dette lag bliver fodret med den nuværende værdi i inputssekvensen, x_t, samt outputsværdien af det forrige stadie, h_{t-1}, her afgøres om cellestadiet skal beholde tidligere information eller glemme det. Outputtet af "forget"-porten f for tidsskridt t findes ved:

$$f_t = \sigma(b_f + W_f \times [h_{t-1}, x_t]) \tag{3.39}$$

hvor b_f og W_f er henholdsvis bias og vægte i "forget"-porten. Outputtet af "forget"-porten f_t er en række værdier fra 0 til 1. For hver værdi i cellestadiet C_{t-1} vil der være en værdi fra outputtet af "forget"-porten, der indikerer i hvilken grad, information skal huskes.
Det næste trin vedrører, hvorledes cellestadiet skal opdateres med ny information. Den nye information, der videresendes til cellestadiet, består af en kombination af output fra "input"-porten og et tanh-lag. Outputtet fra "input"-porten i_t anvender en Sigmoid-funktion, baseret på data fra forrige stadies værdi h_{t-1} og den nuværende inputsvariabel x_t, som i følgende:

$$i_t = \sigma(b_i + W_i \times [h_{t-1}, x_t]) \tag{3.40}$$

hvor σ denoterer en Sigmoid-funktion, b_i er et bias, og W_i er en vægtmatrice. Outputtet i_t er en vektor med værdier, som bestemmer, hvilke værdier der skal opdateres i cellestadiet C_t. Simultant er der et tanh-lag, hvor der benyttes en tanh-aktiveringsfunktion til at transformere samme informationssæt i samme stil som ligning (3.40) som i følgende:

$$\tilde{C}_t = tanh(b_c + W_c \times [h_{t-1}, x_t]) \tag{3.41}$$

Dette tanh-lag udregner en vektor, $\tilde{C}_t$, der består af nye potentielle værdier, som skal med i cellestadiet. Outputtet af "input"-porten, i_t, og tanh-laget, $\tilde{C}_t$, kombineres ved at gange vektorerne sammen, hvilket resulterer i en vektor bestående af nye værdier, der vil blive tilføjet til cellestadiet C_t.
Cellestadiet C_t opdateres ved at anvende outputtet fra "forget"-porten f_t, "input"-porten i_t og de potentielle nye værdier, der skal tilføjes til cellestadiet $\tilde{C}_t$ som i følgende:

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{3.42}$$

Den første del af ligningen fjerner gammel information ved at multiplicere outputtet af "forget"-porten med det forrige cellestadie. Herefter tilføjes ny information til stadiet ved at tilføje outputtet $i_t \times \tilde{C}_t$. Cellestadiet er nu blevet opdateret og sendes videre i neuronen til næste tidsskridt.

Det sidste step i den skjulte neuron i LSTM-netværket er at beregne det skjulte stadie h_t, altså outputtet af LSTM-laget ved det pågældende tidsskridt. Det skjulte stadie er en filtreret version af cellestadiet. Her benyttes igen et Sigmoid-lag til at bestemme, hvilke værdier af cellestadiet, der ønskes berørt. Dette lag kaldes for "output"-porten og outputtet heraf, o_t, er ligeledes en vektor med værdier fra 0 til 1. Værdierne fra cellestadiet bliver transformerede ved brug af en tanh-aktiveringsfunktion og ganges herefter med outputtet af "output"-porten for at filtrere cellestadiet til et ønsket output:

$$o_t = \sigma(b_o + W_o \times [h_{t-1}, x_t]) \tag{3.43}$$

$$h_t = o_t \times tanh(C_t) \tag{3.44}$$

Hermed er alle operationer tilknyttet et tidskridt i den skjulte neuron i LSTM-netværket blevet beregnet. De tre porte, der alle anvender en Sigmoid-aktiveringsfunktion på information fra det tidligere skjulte stadie og det nuværende input, er med til at bestemme, hvilken information der er relevant for det nuværende tidsskridt og kan herved filtrere unødvendig information og fokusere på det relevante. Information fra dette stadie bliver da passeret videre til næste tidsskridt i sekvensen, hvori samme procedurer foretages. Ovenstående gennemgang af en LSTM-neuron tager udgangspunkt i en sekvens-til-sekvens model. Ved en sekvens-til-én opsætning vil der kun være et output ved det sidste tidsskridt. Ved konstant at frasortere unødvendig information undgår LSTM-netværk i en højere grad problemet med forsvindende- eller eksploderende gradienter, og vægte kan herved ændres korrekt under træning af netværk.

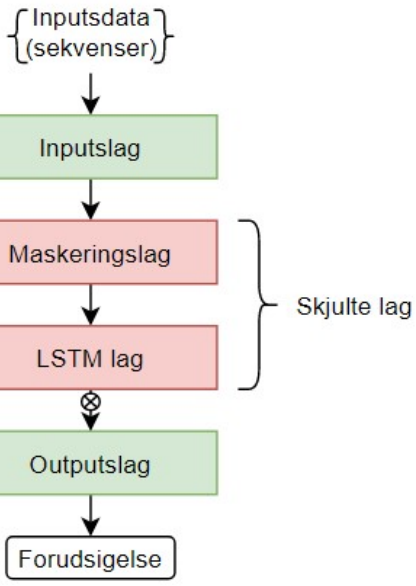
4. Modellering

Følgende afsnit præsenterer arkitekturen og inputsdataet til de kunstige neurale netværks modeller til forecasting af det fremtidige ugentlige afkast for hver af de ti aktiver. Der eksperimenteres med tre netværk pr. aktie, som træner på forskellige frekvenser og længder af tidssekvenser på historisk kursdata. Alle modellerne forecaster den kommende uges kursafkast, som efterfølgende anvendes i et MVO-framework til sammensætning af investeringsporteføljer i en 27 ugers out-of-sample periode. Til dette forecastes variansen og kovariansen mellem de ti aktiver via en DCC-GARCH til anvendelse som risikomål for den repræsentative mean-variance investor. Porteføljerne reallokeres ugentligt hver onsdag, og den økonomiske værdi af investeringsstrategierne, baseret på forecastene af LSTM-modellerne, evalueres til sidst via forvaltningsafgiter.

4.1 Model Arkitektur

Dette projekts LSTM-netværk har to skjulte lag udover inputs- og outputslaget. Det første skjulte lag er et maskeringslag, der maskerer inputssekvenserne ved at springe visse tidsskridt over. I dette projekts tilfælde anvendes det til at håndtere manglende data. Outputtet af maskeringslaget er en sekvens af samme dimension som inputssekvensen. Det andet skjulte lag i netværkene er opbygget efter en LSTM-struktur, og laget har 100 LSTM-neuroner, hvor der anvendes en tanh-aktiveringsfunktion. Outputslaget, der kommer med den endelige forudsigelse, har én neuron og anvender en lineær aktiveringsfunktion. Der er altså tale om en sekvens-til-én opbygning, hvor hver sekvens anvendes til at lave én forudsigelse. Arkitekturen er illustreret i figur (4.1), hvor de skjulte lag er fremhævet.
Mellem LSTM-laget og outputslaget er der placeret et dropout-lag, hvor output fra LSTM-laget tilfældigt droppes. Dette lag tvinger outputtet af et netværks lag til en sparsommelig repræsentation, hvilket medfører, at efterfølgende lag ikke lærer alt for specifikke sammenhænge, som kun findes i træningsdataet.

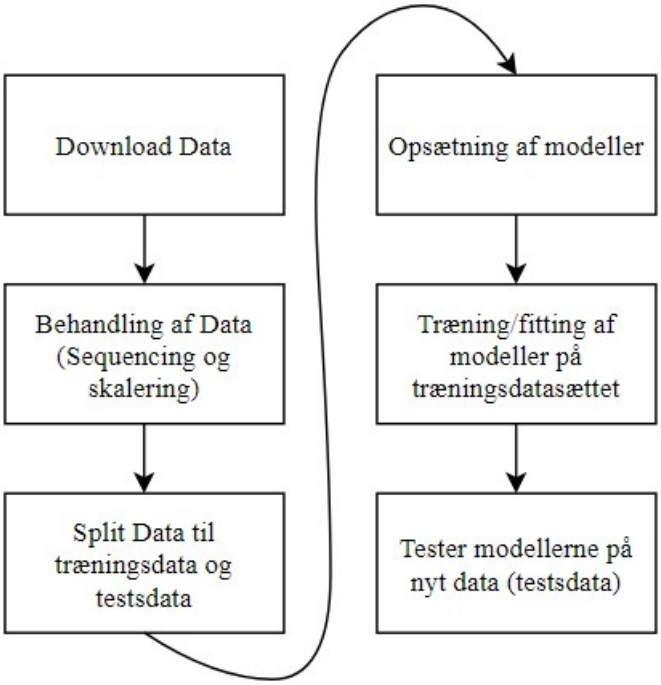
Fig. 4.1: Arkitektur af LSTM-netværket



Der er blevet opsat tre netværk pr. aktie: to intradags modeller, der anvender 3- og 5-dags sekvenser af intervaller på fem minutter og en, der anvender en sekvens af 90 dage med daglige ændringer. Dimensionen af inputslaget afhænger derfor af det pågældende netværk grundet dataets struktur. Den afhængige variabel, som hvert netværk forsøger at forecaste, er den ugentlige procentvise ændring i aktiekursen fra den pågældende sekvens' slutobservation.

Alle parametre, som opsættes inden netværket trænes, kaldes for hyperparametre. Hyperparametre i et netværk kan essentielt deles op i to grupper - den ene gruppe af parametre afgør, hvordan et netværk skal trænes og den anden gruppe fastsætter, hvordan selve netværket er opbygget. Hyperparametre, der er med til at bestemme strukturen, er blandt andet antallet af neuroner og skjulte lag, den anvendte aktiveringsfunktion i de forskellige lag samt andelen af output fra neuroner i et lag, der skal droppes. Til træningen af netværket opsættes hyperparametre som antallet af epoker, batch størrelse og læringsraten. Antallet af epoker afgør, hvor mange gange optimeringsalgoritmen køres for at minimere loss-funktionen, og læringsraten afgør, i hvor høj en grad vægtene og bias i netværket skal justeres efter hver epoke.
Tuningen af disse hyperparametre har til formål at sikre, at modellen ikke blot har høj præcision på træningsdataet men også på nyt og uset data. Undlades tuning af hyperparametrene, kan det medføre, at netværket overfitter træningsdataet og derfor lærer alt for specifikke sammenhænge, der kun gør sig gældende i træningsdataet. Det er derfor et spørgsmål om at opsætte en model, der har kompleksitet nok til at forklare komplekse sammenhænge i dataet uden at være så kompleks, at den under træningen lærer alt for unikke ting, som ikke er generaliserbare til ny data. Den mest simple tuningsmetode er trial and error. Til at starte med opsættes et netværk, hvor hyperparametre sættes relativt arbitrært, herefter trænes netværket. Netværkets præstation på valideringssættet anvendes til at evaluere de initielt opsatte hyperparametre. Efterfølgende justeres de forskellige parametre for finde netværket med den højeste præcision på valideringssættet. Netværkets præstation på træningssættet anvendes til at se om netværket overfitter træningssættet. Det er baseret på denne metode, at strukturen af ovenstående LSTM-netværk er blevet fundet. Netværk med flere LSTM-lag er blevet testet, hvor det ikke blev fundet, at modellen præsterede bedre. Endvidere havde disse netværk en større tendens til at overfitte. Den endelige arkitektur er derfor en relativ sparsommelig model for at modvirke overfitting af træningssættet.
Arbejdsprocessen bag opsætningen af projektets neurale netværk kan opsummeres med figur 4.2.

Fig. 4.2: Neurale netværk arbejdsproces

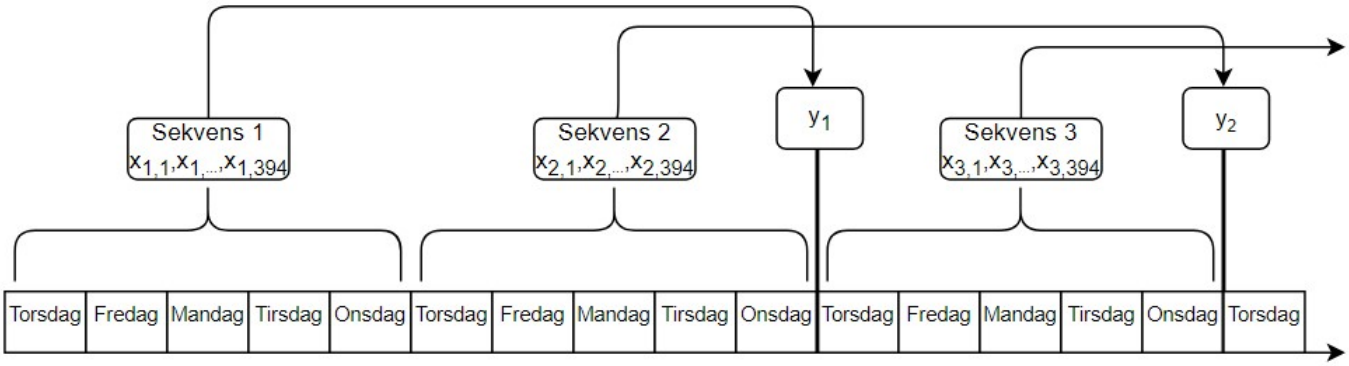


4.1.1 Inputsdata

De neurale netværk med intradagsdata som input anvender 3 til 5 dage af højfrekvensdata mellem kl 9:30-16:00 til at lave deres forecast - det vil sige dage bestående af maksimalt 79 datapunkter hver. Sekvenserne anvender ikke den sidste tilgængelige observation, eftersom porteføljen sammensættes lige inden markedets lukning, hvormed den sidste observation ikke kan være kendt. Der anvendes altid maksimalt fem handelsdage af data, hvilke kan variere i længde, eftersom der f.eks. kan være handelsstop, eller det kan være en halvdag på markedet, hvor almindelig handel stopper kl. 13:00. Dette resulterer i, at nogle af sekvenserne mangler op til 38 datapunkter. De neurale netværk forventer input af lige længder, hvorfor der udføres *padding* af sekvenserne. Dette indebærer at erstatte de manglende værdier i sekvenser med mindre end 236 eller 394 datapunkter med en værdi, der har 0 % sandsynlighed for at opstå i datasættet organisk. Herefter kan tidsskridt med denne værdi ignoreres vha. modellernes maskeringslag. Eftersom lukkekursen onsdag primært er væsentlig i forbindelse med porteføljesammensætningen, udføres træningen med alle ugens dage som afhængig variabel. Sekvenserne i træningsdataet genereres derfor med et rullende tidsvindue af fem dages handelsdata, hvilke anvendes til at forecaste afkastet fem handelsdage frem, hvormed antallet af sekvenser og observationer af den afhængige variabel femdobles. Inputsdimensionerne af

træningsdataet bliver derfor (359, 236, 1) og (359, 394, 1)[1], for henholdsvis 3 og 5-dages sekvenser. Datasættet med daglige observationer bruges til at opstille 90-dages sekvenser, hvoraf der er tilgængelig data siden år 2000, hvilket resulterer i inputsdimensionerne (1053, 90, 1).
I test datasættet vil sekvenser altid strække sig til anden sidste kursopdatering onsdag. Denne sekvens bruges herefter til at forecaste det ugentlige aktieafkast i procent, dvs. fra lukkekurs onsdag til næste uges lukkekurs onsdag[2]. Opsætningen for 5-dages sekvenser kan ses illustreret i figur 4.3.

Fig. 4.3: 5-dags sekvenser og deres korresponderende afhængige variable



Vanishing gradient og sekvenslængder

En udfordring ved recurrent neurale netværk er, som nævnt i forgående teoriafsnit, vanishing gradient problemet. Problemet kan især opstå ved anvendelse af lange sekvenser, da gradienterne af vægte og bias i forhold til loss-funktionen vil blive backpropagated over flere tidsskridt. Essentielt kan det medføre, at gradienterne bliver meget små, herved står det ikke klart, hvilken retning vægte og bias skal ændres for at minimere loss-funktionen. LSTM-modellen blev udarbejdet for at bekæmpe dette problem ved at introducere flere porte inde i selve cellestadiet, som illustreres i figur 3.8. LSTM-netværket kan dog stadig opleve samme problem grundet aktiveringsfunktionerne anvendt af portene i selve cellestadiet (Li et al., 2018). Såfremt sekvenslængderne er for lange, kan vanishing gradient problemet stadig opstå. Hvis problemet er relevant kan kortere sekvenser være gavnlige for netværket. Det vælges derfor også at konstruere et neuralt netværk, hvor sekvenserne blot består af tre handels-

[1] Dimensionerne læses som: (Antal sekvenser, tidsskridt i sekvenserne, antal features)

[2] Forecast fra onsdag til onsdag er valgt for ikke at overraske netværkene med kursudsving, der kan ske over weekenden, altså uden for handelsperioden.

dage. Det teoretiske grundlag for tre-dages sekvenser er svagere end fem-dages sekvenser, eftersom der, alt andet lige, må være færre signaler i tre dages handel end fem.
Der vil ligeledes også blive konstrueret et tredje netværk, der ikke anvender intradagsdata men i stedet træner på sekvenser af daglige kursobservationer. Dette forsøges for at øge antallet af observationer, netværket kan træne på[3]. Til at teste det neurale netværk med dagligt data anvendes sekvenser bestående af 90 dages handel til at forecaste en uge frem. Med data siden år 2000 kan der opstilles 1053 sekvenser i træningssættet, hvilke anvendes til at kalibrere netværkets vægte. Testsættet anvender de resterende 27 sekvenser til at forecaste de samme datoer som de andre to netværk.

Datasplit

I alt består højfrekvens datasættet af 101 uger af data, hvilke opdeles i et træningssæt og et testsæt. LSTM-modellerne trænes på i alt 74 ugers data, frem til onsdag den 7. oktober 2020, og de resterende 27 uger bruges til at lave de forecast, der anvendes til at konstruere projektets porteføljer. Ydermere anvendes 19 tilfældige uger af træningsdata per epoke til et valideringssplit, hvor modellen kan evalueres i forbindelse med hyperparameter tuning. Den bedste model vælges baseret på dennes præstation på validationssættet, hvilken bruges med testsættet til at lave forecast. Efter dette punkt må der ikke ændres på modellens hyperparametre for således at simulere modellens ydeevne på nyt data. Den første værdi, der forecastes, er lukkekursen den 21. oktober 2020. Hertil anvendes kursdata for den respektive aktie fra torsdag den 8. oktober til onsdag den 14. 2020 som input til modellen.

4.1.2 Univariate GARCH og DCC-GARCH

De univariate GARCH-modelspecifikationer vælges ved brug af Ljung-Box testen på op til fire lags, eftersom alle ARCH-effekter ønskes forklaret, inden disse anvendes i DCC-modellen. Derfor fittes forskellige modelspecifikationer på hele tidsserien for alle aktier. Her vælges den mest sparsommelige modelspecifikation, der består Ljung-Box testene på et 10 % signifikansniveau. Dette viste sig at være en *ARMA(1,1)-EGARCH(1,1)* for alle aktier pånær \$CAT, hvor en *ARMA(1,1)-EGARCH(2,1)* bestod Ljung-Box testene. Ved at anvende en *Exponential* GARCH-specifikation, muliggøres det at modellere asymmetrisk volatilitet, hvilket tyder på at være relevant for projektets aktier. De univariate GARCH-modeller estimeres og anvendes af projektets DCC(1,1)-model, hvilken bruges til at forecaste kovariansmatricen én uge frem.

[3] Argumentet for at øge antallet af observationer er diskuteret i afsnit 5.2.2

Datasættet til volatilitetsmodellering består af et rullende tidsvindue af 557 ugentlige observationer. Dette data anvendes til at forecaste varians-kovariansmatricen én uge frem. Herved anvendes al data fra starten af datasættet, indtil den første portefølje skal sammensættes, til at lave det første forecast af aktiernes kovarianser. Herefter vil det rullende tidsvindue konstant fremrykkes med én uge pr. forecast. DCC-GARCH-modellen fittes i alt 27 gange, hvor hver model benyttes til at lave et forecast.

4.2 Model resultater

De tre neurale netværk trænet på henholdsvis 3- og 5-dages sekvenser af intradags data samt 90 handelsdages daglige afkastdata[4], vil blive sammenlignet med en random walk model og et simpelt historisk gennemsnit, udregnet ud fra det ugentlig afkast for de sidste 30 handelsuger. Random walk-modellen forecaster blot næste uges fremtidige afkast som denne uges observerede afkast. Evalueringerne vil udelukkende foregå på out-of-sample data, for at se hvordan modellerne præsterer på uset data.
Tabel 4.1 viser Root Mean Squared Error (RMSE) for out-of-sample forecast for de tre neurale netværk af alle ti akter samt for en random walk-proces og et historisk gennemsnit for de seneste 30 handelsuger.

Tabel 4.1: Out-of-sample Root Mean Squared Error (RMSE)

	3-dag	5-dag	Daglig	RW	Historisk
BA	9.30%	10.28%	6.92%	8.24%	6.59%
CAT	5,70%	5,84%	4,87%	4.42%	6,73%
DIS	5,15%	5,44%	5,38%	4,68%	6,42%
IBM	4,15%	3,78%	4,25%	4,07%	5,43%
JPM	5,35%	4,88%	4,72%	4,25%	5,52%
MCD	3,43%	5,53%	2,95%	2,44%	3,32%
MRK	3,04%	2,96%	3,06%	2,66%	3,68%
MSFT	3,27%	3,21%	3,69%	3,16%	4,40%
PG	2,41%	2,61%	1,96%	2,11%	2,53%
WMT	3,72%	4,28%	3,30%	2,85%	3,60%
Gennemsnit	4,55%	4,88%	4,11%	3,72%	4,99%

[4]Vil fremadrettet blive henvist til som "Daglig"-modellen.

Som det fremgår af tabellen, har alle modellerne en meget høj RMSE for forecastene af Boeings aktiekurs (BA). Denne har været den klart mest volatile aktie i både trænings- og testperioden, og ingen af de kunstige neurale netværk har altså lært at akkurat modellere denne volatilitet. Af de tre LSTM-modeller, har Daglig-modellen generelt haft den mindste fejlmargin i forecastene på out-of-sample perioden. Dog forudsiger random walk modellen næste uges kursafkast med mindre fejlmargin, end de tre LSTM-modeller formår. Baseret på RMSE, udkonkurrerer de kunstige neurale netværk ikke et mere naivt forecast i form af en random walk.
RMSE alene kan give misvisende billeder af, hvordan en model præsterer. Et andet mål, der kan anvendes til sammenligning af forecast, er modellernes *Mean Directional Accuracy* eller MDA. I dette tilfælde anvendes en "blødere" definition på en retningsbestemmelse, hvor det fastslås, at det ikke er fortegnet af afkastet men størrelsesforholdet af de forventede afkast, der definerer retningen. Det vil sige at hvis det forventede afkast er lavere (men ikke nødvendigvis negativ) end forrige periodes forecast, og det samme er gældene for det faktiske afkast, anses det indenfor denne definition som en korrekt retningsbestemt forudsigelse.

Tabel 4.2: Out-of-sample Mean Directional Accuracy (MDA)

	3-dag	5-dag	Daglig	RW	Historisk
BA	48,1%	40,7%	29,6%	25,9%	44,4%
CAT	51,9%	48,1%	44,4%	37,0%	33,3%
DIS	37,0%	37,0%	22,2%	37,0%	25,9%
IBM	48,1%	51,9%	18,5%	25,9%	33,3%
JPM	37,0%	37,0%	37,0%	33,3%	37,0%
MCD	51,9%	51,9%	37,0%	25,9%	33,3%
MRK	37,0%	40,7%	25,9%	33,3%	37,0%
MSFT	44,4%	48,1%	25,9%	44,4%	48,1%
PG	40,7%	40,7%	33,3%	33,3%	33,3%
WMT	48,1%	40,7%	29,6%	33,3%	29,6%
Gennemsnit	44,4%	43,7%	30,4%	33,0%	35,6%

Som det fremgår i tabel 4.2, præsterer LSTM-modellerne trænet på intradagsdata bedre end Daglig-modellen og de to naive forecastingsmodeller. 3- og 5-dags modellerne har en gennemsnitlig retningsbestemt nøjagtighed på henholdsvis 44,4% og 43,7%, hvilket stadig betyder, at i over halvdelen af tilfældende over out-of-sample perioden forecaster disse modeller ikke korrekt, om afkastet var lavere eller højere end den forrige periodes aktieafkast.

Resultaterne indikerer, at netværk trænet på intradagsdata har været i stand til at fange et signal i inputsdataet, som forbedrer modellernes retningsbestemte forudsigelse - dog ikke op til et imponerende niveau.

Pesaran-Timmermann test

Endnu en evaluering foretaget på modellerne er en test foreslået af Pesaran & Timmermann (1992) kendt som Pesaran-Timmermann testen. Pesaran-Timmermann testen evaluerer statistisk, hvorvidt en model, med forecast x, er i stand til at forudsige ændringen i retningen af en tidsserie y. Under nulhypotesen om, at x ikke forecaster retningen af ændringen i y, kan følgende test statistik anvendes:

$$S_n = \frac{\hat{P} - \hat{P}_*}{\sqrt{\hat{\text{var}}(\hat{P}) - \hat{\text{var}}(\hat{P}_*)}} \sim N(0,1) \tag{4.1}$$

hvor

$$\hat{P} = n^{-1} \sum_{t=1}^{n} Z(y_t x_t)$$

$$Z(\cdot) = \begin{cases} 1 & \text{hvis } \cdot > 0 \\ 0 & \text{ellers} \end{cases}$$

$$\hat{P}_y = n^{-1} \sum_{t=1}^{n} Z(y_t) \quad \hat{P}_x = n^{-1} \sum_{t=1}^{n} Z(x_t)$$

$$\hat{\text{var}}(\hat{P}) = n^{-1} \hat{P}_*(1 - \hat{P}_*)$$

$$\hat{\text{var}}(\hat{P}_*) = n^{-1}(2\hat{P}_y - 1)^2 \hat{P}_x(1 - \hat{P}_x) + n^{-1}(2\hat{P}_x - 1)^2 \hat{P}_y(1 - \hat{P}_y) + 4n^{-2}\hat{P}_y\hat{P}_x(1 - \hat{P}_y)(1 - \hat{P}_x)$$

Resultaterne af Pesaran-Timmermann testen på de tre neurale netværks modeller fremgår i tabel 4.3.

Tabel 4.3: Out-of-sample Pesaran-Timmermann test-statistikker

	BA	CAT	DIS	IBM	JPM	MCD	MRK	MSFT	PG	WMT
3-dags	-1,674	-0,796	0,227	-0,765	-1,522	-1,001	0,082	-0,316	-0,965	-0,861
5-dags	-0,79	—	-0,416	0,989	-1,101	—	0,056	0,53	-1,187	—
Daglig	1,404*	-1,433	-0,353	0,282	1,203	-1,609	-0,562	—	0,176	-0,861

*Signifikante teststatistikker på et 10 % signifikansniveau markeres med *, 5 % signifikansniveau med **, og et 1 % signifikansniveau med ***. — = udefinerbar*

Resultaterne af Pesaran-Timmermann testene på de tre neurale netværks modeller forkastes i alle undtagen et enkelt tilfælde. I enkelte tilfælde forudsiger modellerne udelukkende positive afkast, hvorfor PT-testen ikke er definerbar for disse modeller, eftersom de slet ikke forudsiger et retningsskift i den forudsagte tidsserie. Det kan altså ikke påpeges, at de estimerede forecastmodeller akkurat er i stand til at forudsige retningsskift for aktieudviklingen på de ti aktier i fokus.

DCC-GARCH

For at vurdere om DCC-GARCH-modellen er passende til at forecaste projektets kovariansmatricer, udføres signifikanstests på a- og b-parametrene, der ses i ligning (3.19). Såfremt disse er signifikante, kan det ikke afvises, at de betingede korrelationer varierer. Tabel 4.4 indeholder et uddrag af de 27 ugers estimater[5].

Tabel 4.4: Uddrag af DCC-GARCH a og b parametre

Uge	a	b
1	0.011*	0.705***
5	0,012**	0,713***
10	0,012*	0,646***
15	0,012**	0,668***
20	0,011*	0,669***
25	0,010	0,644***
27	0,010	0,645***

*Signifikante teststatistikker på et 10 % signifikansniveau markeres med *, 5 % signifikansniveau med **, og et 1 % signifikansniveau med ***.*

Samtlige 27 test af b påviste at disse var signifikant anderledes fra 1, på et 1 % signifikansniveau, og 24 af a var signfikant anderledes fra 0 på et 10 % signifikansniveau, hvormed det konkluderes at det er relevant at modellere de betingede korrelationer som dynamiske.

4.3 Investeringsstrategi

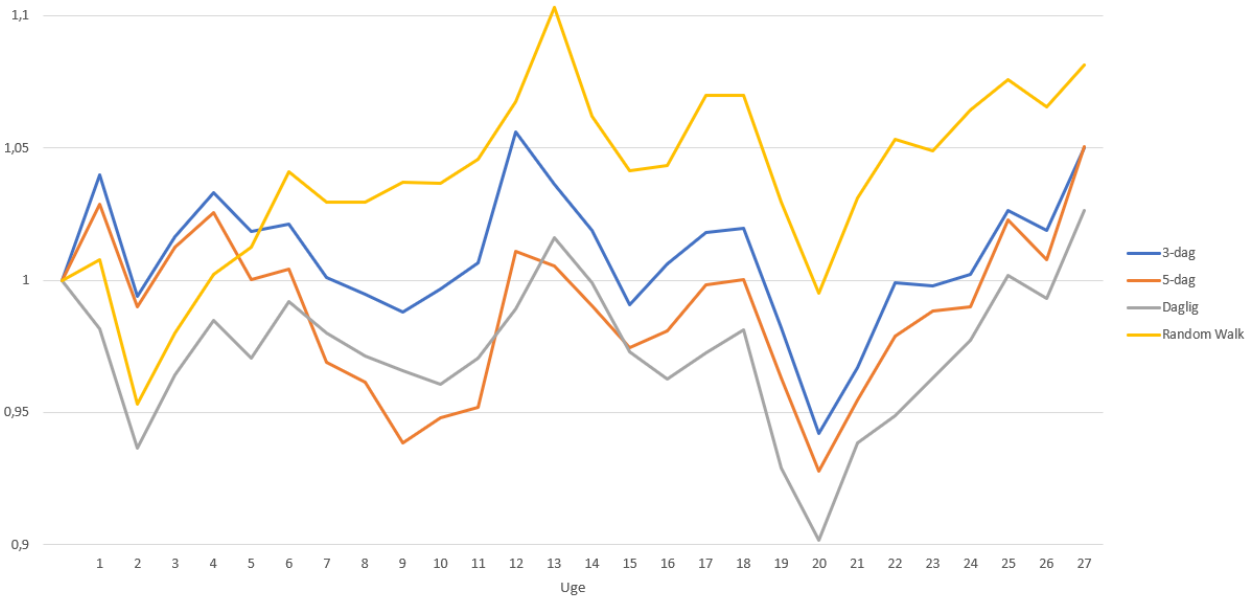
Forecastene på out-of-sample dataet vil blive anvendt i en dynamisk porteføljesammensætning. De dynamiske porteføljevægte re-allokeres ugentligt hver onsdag, lige før markedet lukker

[5]Der vises ikke alle 27 uger, eftersom estimaterne ikke varierer meget i det rullende tidsvindue.

ved at løse porteføljeoptimeringsproblemet beskrevet i ligning (3.5). Der opsættes i alt 27 porteføljer, svarende til de 27 forecasts af afkast og varians-kovarians-matricer. Den første og sidste portefølje opsættes henholdsvis den 14. oktober 2020 og 14. april 2021. Der anvendes en repræsentativ investor, hvis risiko aversitets-parameter γ holdes fast over hele investeringsperioden[6].

Figur 4.4 illustrerer værdien af en investeret enhed, hvis porteføljesammensætnings-strategien følges over den 27 ugers out-of-sample periode, baseret på forventninger givet af de tre LSTM-modeller, samt porteføljen sammensat vha. random walk forecasts.

Fig. 4.4: Kumulativ porteføljeværdi over out-of-sample investeringsperioden



Generelt følger udviklingen af de to modeller trænet på intradagsdata hinanden, men strategien baseret på 3-dags sekvenser præsterer under hele perioden bedre end 5-dags modellen. Dette ændrer sig dog på den sidste uge af investeringsperioden, hvor 5-dags modellen ender med et marginalt højere samlet afkast over perioden. På trods af, at modellen trænet på daglige kursobservationer generelt havde en lavere RMSE på de ti aktiver, præsterer porteføljerne sammensat på disse forecast værst i investeringsperioden. Investeringsstrategien, der følger et naivt forecast baseret på en random walk, fremgår også i figur 4.4. Random walk porteføljerne præsterer bedre end sammensætningen baseret på de tre LSTM-modellerede afkast. Kun den første måned præsterer netværkene trænet på intradagsdata

[6]Vi fastholder en arbitrær værdi på $\gamma = 250$, eftersom dette giver rimelige forventede afkast på porteføljesammensætningerne givet ugentlige investeringshorisonter. Det er værd at pointere, at γ ikke er den samme som den absolutte risikoaversitets parameter λ i ligning (3.7), da γ skal ses i kontekst af en mean-variance investor, som er avers overfor porteføljens varians.

bedre end random walk porteføljerne, hvor denne klarer sig bedst perioden ud. Det er interessant at pointere, at porteføljeværdien, baseret på de tre modeller, generelt følger udviklingen for den naive portefølje. For 3-dags og 5-dags porteføljerne skyldes det lave afkast over perioden, sammenlignet med random walk porteføljerne, hovedsageligt præstationen fra uge 6 til 9, hvor de modelbaserede porteføljer oplever tab, hvorimod random walk porteføljen generelt vedligeholder sin værdi.
Som tidligere beskrevet fejler modellerne Pesaran-Timmermann testen for alle aktiverne. Det er da ikke overraskende, at porteføljerne baseret på LSTM-modellernes forecast ofte kan opleve negative afkast. Investeringsperioden er meget volatil med store kursudsving for de ti aktiver, som ofte skifter mellem negativ og positiv. Eftersom modellerne ikke har vist sig at være gode til at forudsige retningsskift i kursudviklingen, vil porteføljesammensætningen baseret på disse forecast ofte medføre tab.

4.3.1 Økonomisk evaluering

Tabel 4.5 indeholder porteføljestrategiens afkast i perioden, μ, og den realiserede volatilitet i perioden, σ, for porteføljerne sammensat ved brug neurale netværk. Ydermere udregnes porteføljeafgiften[7], Φ, en investorer maksimalt vil betale for at skifte fra porteføljen sammensat med en random walk model. Afgiften findes ved at sætte nytten af de sammenlignede porteføljer lig hinanden baseret på en repræsentativ investor med en kvadratisk nyttefunktion, som beskrevet i afsnit 3.2.
Her henviser Φ_1 og Φ_{10} til porteføljeafgifter for en repræsentativ investor med en fast relativ risikoaversitetsfaktor δ med henholdsvis en værdi på $\delta = 1$ og $\delta = 10$.

Tabel 4.5: Opsummerende statistikker for porteføljer

	μ	σ	Φ_1	Φ_{10}
3-dag	5,01%	12,55%	-10,76	-13,21
5-dag	5,05%	13,27%	-10,62	-15,74
Daglig	2,64%	12,23%	-19,19	-20,38

Porteføljernes afkast i den 27 ugers periode var små i forhold til de realiserede volatiliteter[8]. Porteføljernes nytte var lavere end benchmark-porteføljen og som følge heraf, var alle Φ-værdier negative. Det må altså konkluderes, at porteføljessammensætningen, baseret på LSTM-modellernes forecast, ikke har formået at overgå et simpelt naivt forecast via en

[7] Afgiften Φ er en andel af den samlede formue investeret og rapporteres i basispoint.

[8] Porteføljeafkastet måles foruden de handselsudgifter, der typisk findes på de finansielle markeder, hvilke, igennem 27 rebalanceringer, ville formindske afkastet.

random walk model. Baseret på dette evalueringskriterie har porteføljestrategierne ikke skabt ekstraordinær økonomisk værdi i testperioden.

5. Diskussion

5.1 Kritik af Mean-Variance frameworket

Mean-variance optimization (MVO) frameworket har været, og er stadig, meget populært anvendt på grund af dets egenskab til at implementere to af de mest fundamentale punkter indenfor porteføljesammensætning, diversifikation og afvejningen mellem forventet afkast og risiko i et analytisk medgørligt framework. Denne er dog ikke uden kritik. Frameworket bygger implicit på en antagelse om, at investoren har en kvadratisk nyttefunktion (Samuelson, 1967, Borch, 1969, Feldstein, 1969). Resultatet af sådan en nyttefunktion vil være, at investoren i stigende grad bliver mere risikoavers i takt med, at investorens velstand stiger (Arrow, 1965, Pratt, 1964). Dette er teoretisk uattraktivt og går imod hvad, der empirisk observeres[1].

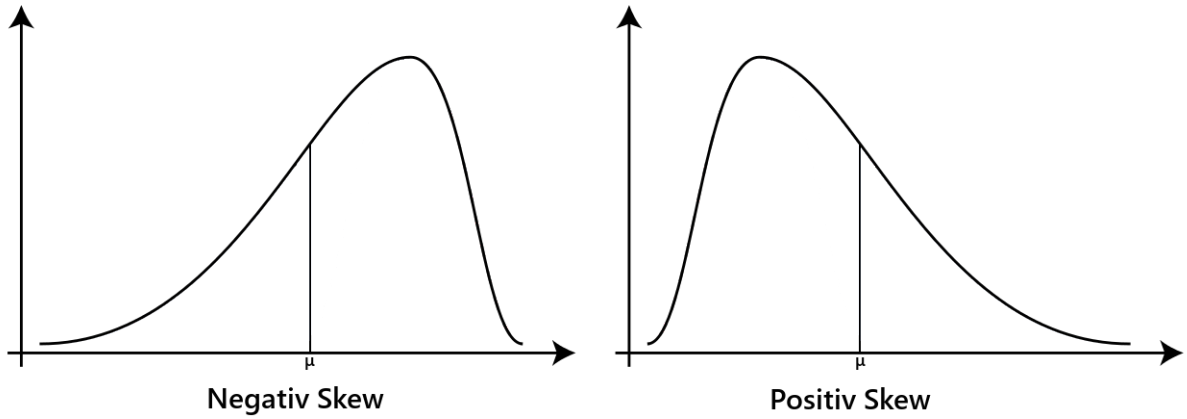
Relateret til dette er et andet punkt, der kritiseres, fordi det medfører, at mean-variance frameworket kun tager højde for de første to momenter; middelværdi og varians, og ignorerer højere ordens momenter for afkastet. Mange påpeger, at det tredje og fjerde moment, det som er kendt som *skewness* og *kurtosis*, ikke kan anses som økonomisk ubetydelige og bør tages højde for i en "optimal" porteføljesammensætning[2]. Kritikken går her på, at de første to momenter; middelværdi og varians, kun er tilstrækkelige, hvis fordelingen af afkast på aktiverne er normalfordelte, eller investeringshorisonten er meget kort (Samuelsen, 1970). Dette er dog ikke, hvad der empirisk er observeret. Arditti (1967), Fielitz (1976), Simkowitz & Beedles (1978) og Singleton & Wingender (1986) finder alle, at aktieafkast hyppigt udviser positiv skewness. En investor bør netop tage denne asymmetri af afkastene med i sin vurdering om det fremtidige forventede afkast og risikovurdering af investeringen. En risikoavers investor vil i teorien se positivt på positiv skewness og omvendt være varig for investeringer, som måtte udvise negativ skewness i afkastdistributionen. Udviser afkastet for en investering positiv skewness, betyder det, at der pålægges større sandsynlighed for, at udsving i afkastet fra forventningen vil være i mindre størrelsesordner. Alt andet lige vil positiv skewness betyde en mindre risikabel investering. Omvendt vil negativ skewness

[1] Se f.eks. Cohn. et. al. (1975) eller Guiso & Paiella, (2008)

[2] Se f.eks. Samuelsen (1970), Scott & Horvath (1980), Kane (1982), Lai (1991) eller Athayde & Flores (2004)

betyde, at en investering udviser større sandsynlighed for høj volatilitet i afkastet, hvilket, man må antage, er uattraktivt for en risikoavers investor.

Fig. 5.1: Skewness distribution illustration



En investeringssammensætning, der kun tager højde for de to første momenter, vil ikke kunne anses som optimal, hvis afkastet på aktiverne udviser skewness i distributionen i en ikke-ubetydelig størrelsesorden. Værdisætter investoren positiv skewness, eller ligeledes kræver en risikopræmie for at påtage sig negativ skewness, bør dette inkorporeres i investeringssammensætningen. Simkowitz & Beedles (1978) finder, at sammensætningen og diversifikationen af en portefølje, hvor der ikke også tages højde for positiv skewness i afkastene, medfører, at denne positive egenskab forsvinder. Sådan en portefølje vil netop ikke være optimal for investoren, som værdisætter positiv skewness.

Udfordringen ved at inkorporere højere momenter, som antageligt er teoretisk valide, er, at en analytisk løsning til, hvordan en investor optimerer sammensætningen af sin investeringsbeslutning, er kompleks og svær at definere[3]. Singleton & Wingender (1986) finder også, at aktieafkast hyppigt udviser positiv skewness men finder dog, at denne ofte ikke er vedvarende. Heraf udtrykker de bekymring om forudsigelse af fremtidig skewness i afkast baseret på historisk data. Arditti & Levi (1975) påpeger, at vigtigheden af skewness meget vel kan afhænge af investeringshorisonten, som en given investor arbejder med. Arditti & Levi viser, at jo længere en investeringshorisont, jo større en effekt kan skewness komme til at have, og det bliver da af stigende betydning at tage højde for dette højere moment for en investors investeringsbeslutning. Omvendt påpeger Arditti & Levi også, at empirisk arbejde fandt distributionen af afkast symmetrisk på kortere sigt, og der derfor på kort sigt er belæg for, at skewness ikke har nogen effekt på en sådan kort investeringshorisont. Dette

[3]Nyere forskning har forsøgt sig med adskillige metoder, som tilnærmelsesvist estimerer en løsning med højere momenter inkorporeret. Se f.eks. Briec et. al (2005) eller Beardsley et. al (2012)

står også overens med Samuelsens (1967 & 1970) argumenter om at MVO-løsningen med to momenter, på tilpas kort investeringshorisont, kan anses som en udemærket approksimation.

5.2 Machine Learning og finans

Når man ønsker at gøre brug af machine learning til kapitalforvaltning, er det vigtigt at forstå problemstillingen, man sætter op og de specifikke udfordringer, der skal tages højde for, hvis man vil anvende neurale netværk indenfor dette område. Dette afsnit vil præsentere nogle af de udfordringer, de finansielle markeder indebærer, som er relevante for anvendelsen af neurale netværk og andre tilhørende aspekter, som investorer bør tage højde for.

5.2.1 Lav signal-to-noise ratio

Implementering af machine learning teknikker inden for den finansielle verden er ikke uden udfordringer. Moderne machine learning teknikker trives i datamiljøer, som udviser høj signal-to-noise ratio. Så længe der er et forhold, som udviser stærke signaler til, hvad end problemstilling der arbejdes med, har neurale netværk haft stor succes. Dette skyldes, at det stort set er ligemeget, hvad interaktionen det informationsbærende signal har med den datagenererende proces, og hvilken form interaktionen måtte have. Neurale netværk har ingen forforståelse om den datagenererende proces, men lærer i stedet fra inputsdataets variable for at finde de signaler, som driver udviklingen.
I forbindelse med markedsforcasting er man som regel kun interesseret i korrekt forecasting af risikopræmien. En udfordring ved forecasting af risikopræmier på aktiemarkedet er, at markedsdata udviser lav signal-to-noise ratio (Kelly et. al, 2020). Den lave signal-to-noise ratio opstår på grund af de mest basale markedskræfter inden for profitmaksimering og konkurrence. Findes der information, som pålideligt kan forudsige markedsudviklinger, vil handlende aktører på markedet udnytte den relevante information, indtil værdien af denne er udtømt. Udnyttelsen af informationen medfører, at prisen på det relevante aktiv udvikler sig, indtil det prædiktive signal helt forsvinder, og dette dermed er fuldt inkorporeret i prisen. Denne egenskab af markedet er netop det, den nobel pris-vindende teori, og arbejde af Eugene Fama (1965, 1970, et. al. 1969), *the efficient market hypothesis*, baseres på. Hvis de informationsbærende signaler allerede er priset ind, vil det kun være støj, altså uforudsigelige stød, som bevæger markedet.
Endnu en udfordring ligger i, at de finansielle markeder konstant udvikler sig. Opstår eller identificeres et nyt værdiskabende signal eller introduceres ny teknologi, vil markedskræfterne medføre, at dette nu også bliver inkorporeret i prisfastsættelsen. Dette kan betyde, at

prisudviklingen nu ser anderledes ud for de relevante aktiver, hvilket essentielt svarer til, at den datagenererende proces ændrer sig, grundet aktørerne nu agerer på ny information. Udviklingen af de finansielle markeder gør modelleringsproblemstilling meget mere kompleks sammenlignet med de andre områder, hvor anvendelsen af machine learning har set succes. Dette betyder også, at de neurale netværk konstant skal gentrænes, fordi landskabet løbende ændrer sig. Perioder, hvor en model præsterer udemærket, kan hurtigt skifte forhold, der medfører, at andre eller nye signaler nu er mere gældende for den datagenererende proces.

5.2.2 Kapitalforvaltning og frekvens

For at gøre aktieafkast forecasting til et *big data* problem, er det nødvendigt at hæve frekvensen af datasættet for at øge antallet af observationer (Kelly et. al. 2020). Men ved anvendelse til aktiv allokering af investeringer er man også nødt til at tage højde for frekvensen af reallokering. Udfordringen her ligger i, at der er handelsomkostninger involveret i handel af aktier, og disse stiger i takt med antallet af handler. Disse omkostninger bør der tages højde for, hvis målet er at udnytte en forecastingsmodel i praksis. En problematik opstår dog her. Hvis man i praksis anseende skal anvende denne form for machine learning modellering til aktiv porteføljeforvaltning, kan man ikke påstå, at det er et big data problem længere. Dette skyldes, at det er ugentlige observationer, der skal forecastes på, og antallet af observationer er derfor begrænset af, hvor mange uger dataet består af. Denne type modellering er altså ikke nødvendigvis oplagt til sådan en problemstilling, med mindre det kan påvises, at netværksmodellerne konsistent og i høj nok grad outperformer andre metoder, selv med så få dataobservationer.
Netværkstypen er veloplagt til at estimere over mange parametre, som vi netop har inkorporeret ved at anvende høj-frekvens data. En potentiel udfordring med dette projekts problemstilling er, at det kan argumenteres, at der ikke er nok observationer, som det neurale netværk kan lære af for at fange de potentielle ikke-lineære forhold, som måtte eksistere i dataet. Et mere oplagt område for denne type modellering ville være under høj-frekvens handel af aktiver. Under så kort et tidsinterval mellem handler, vil der være mange gange flere afkastobservationer at lære fra, så længe målet er aktivt at handle på den frekvens. Neurale netværk trives i sådan et *big data* miljø, fordi det neurale netværk kan træne over mange flere observationer og har dermed bedre mulighed for at fange potentielle skjulte signaler, der måtte eksistere i datasættet. Begrænsningen ved denne type trading er, at handelsstørrelsen nødvendigvis er tvunget til at være lille. Ønsker man at opnå fortjeneste på minut-til-minut-handler, vil der kun være en lille fortjeneste, før handlen i sig selv påvirker kursen, og den profitable mulighed forsvinder.

Ovenstående og forrige afsnit illustrerer, at de finansielle markeder, og især afkastforudsigelse, står over for et udfordrende sæt forhold, som adskiller sig fra de andre områder, hvor machine learning har haft succes.

5.2.3 Inkorporering af økonomisk teori

I dette projekt er der kun anvendt én inputsvariabel til at forecaste aktieafkast, den historiske kursudvikling. Dette er gjort på trods af det faktum, at neurale netværk er i stand til at håndtere store datasæt med mange forklarende variable. Modellerne opstillet i dette projekt er derfor meget simple modeller, hvilket kan kritiseres indenfor denne problemstilling. Kelly et al. (2020) argumenterer, at anvendelsen af ustruktureret data i et neuralt netværk til forudsigelse af aktiekursudvikling vil være inefficient brug af data. Som tidligere diskuteret, anses kursforudsigelse som et small data problem, grundet det begrænsede antal observationer indenfor dette område, som et neuralt netværk kan anvende til at lære af. Derudover er der også den førnævnte problemstilling med lav signal-to-noise i kursdata. Disse udfordringer kan dog potentielt behjælpes, hvis man strukturerer inputsdataet ved at inkorporere økonomisk teori. Et eksempel på dette kunne være den anerkendte Arbitrage Pricing-teori, introduceret af Ross (1976). Ross argumenterer, at aktiekurser følger en faktorstruktur, hvor flere faktorer styrer udviklingen af merafkastet af aktiepriser. Hvad disse faktorer måtte være, er vidt diskuteret og kan varierere fra makroøkonomiske variable til f.eks. faktorerne fra den kendte Fama-French tre faktor model (Fama & French, 1993).
Som Kelly et al. (2020) argumenterer, afhænger fremgangsmetoder, der udelukkende anvender ustruktureret data, på en antagelse om, at det neurale netværk i sig selv formår at opfange faktorstrukturen, som måtte eksistere i dataet. Denne antagelse kan da skarpt kritiseres, eftersom dette ikke kan antages med sikkerhed, og er især udfordret i dette projekts problemstilling grundet small data problemet, dette er præget af.

5.2.4 Netværksarkitektur - mere kunst end videnskab

Konstruktionen af arkitekturen af et neuralt netværk sammenlignes ofte mere med kunst end som en eksakt videnskab. Inden for neurale netværk og deep learning findes der mange forskellige modeltyper, og dette projekt anvender blot en af dem (LSTM), fordi denne modeltype teoretisk set er velegnet til tidsseriedata. Af alle disse typer modeller inden for neurale netværk er der nærmest uendeligt mange måder at strukturere arkitekturen af et netværk på. Man kan på forhånd have en hypotese om, hvilken modeltype og struktur vil være mest oplagt og præstere bedst til den givne problemstilling, men der findes endnu ikke en eksakt metode til specificering af netværksstruktur.

Beslutningen om, hvor mange lag, antallet af neuroner, lærings-algoritmer samt de mange andre hyperparametre, der kan justeres på, kommer ofte an på en *trial and error*-tilgang. Dette lægger sig op ad, at kunstige neurale netværk ofte sammenlignes med en sort boks. Denne tilgang er dog nødvendig, da det stort set er umuligt at tolke på, hvad et neuralt netværk præcist lærer under træningen.
Chong et. al. (2017) præsenterer en kort gennemgang af tidligere arbejde med neurale netværk og aktiemarkedet. Et uddrag af disse er vist i tabel 5.2[4]. Heri kan de mange forskellige tilgange og netværksarkitekturer observeres. Bemærk, hvor forskellige disse kan være indenfor det samme emnefelt.

Fig. 5.2: Oversigt over tidligere studier på aktiemarked forudsigelse via neurale netværk

Forfattere	Datatyper (Antal af input variable × lagged tid)	Outputmål	Antal af stikprøver (Træning: Validering: Test)	Tidsperiode (Frekvens)	Metode	Præstationsmål
Enke & Mehdiyev (2013)	US S&P 500 indekset (20 × 1)	Aktiekurs	361	Jan-1980 til Jan-2010 (daglig)	Feature selection + fuzzy clustering + fuzzy NN	RMSE
Niaki & Hoseinzade (2013)	Korea KOSPI200 indekset (27 × 1)	Markedsudvikling (op eller ned)	3650 (8:1:1)	1-Mar-1994 til 30-Jun-2008 (daglig)	Feature selection + ANN	Statistiske test
Cervelló et al. (2015)	US Dow Jones indekset (1 × 10)	Markedstrend (bull/bear-flag)	91.307	22-Maj-2000 til 29-Nov-2013 (15-min)	Template Matching	Trading simulering
Patel et al. (2015)	India CNX og BSE indeks (10 × 1)	Aktiekurs	2393	Jan-2003 til Dec-2012 (daglig)	SVR + (ANN, RF, SVR)	MAPE, MAE, RMSE, MSE
T.-l. Chen & Chen (2016)	Taiwan TAIEX[a] og US NASDAQ[b] indeks (27 × 20)	Markedstrend (bull/bear-flag)	3818[a] 3412[b] (7:0:1)	7-Jan-1989 til 24-Mar-2004 (daglig)	Dimension reduction + template matching	Trading simulering
Chiang et al. (2016)	22 aktieindeks (3~5 × 1)	Trading signal (aktiekurs)	756 (2:0:1)	Jan-2008 til Dec-2010 (daglig)	Particle swarm optimization + ANN	Trading simulering
Chourmouziadis & Chatzoglou (2016)	Greeze ASE general indeks (8 × 1)	Portefølje-sammensætning	3907	15-Nov-1996 til 5-Jun-2012 (daglig)	Fuzzy system	Trading simulering
Qiu et al. (2016)	Japan Nikkei 225 indekset (71 × 1)	Aktieafkast	237 (7:0:3)	Nov-1993 til Jul-2013 (månedlig)	ANN + (genetic algoritm, simulated annealing)	MSE
Arévalo et al. (2016)	Apple aktiekurs (3 × (2~15)+2)	Aktiekurs	19.109 (17:0:3)	2-Sep-2008 til 7-Nov-2008 (1-minut)	Deep NN	MSE, Aktieudvikling (op eller ned)
Zhong & Enke (2017)	US SPDR S&P 500 ETF (SPY) (60 × 1)	Markedsudvikling (op eller ned)	2518 (14:3:3)	1-Jun-2003 til 31-Maj-2013 (daglig)	Dimension reduction + ANN	Trading simulering, statistiske test
Chong et al. (2017)	Korea KOSPI 38 aktieafkast (38 × 10)	Aktieafkast	73.041 (3:1:1)	4-Jan-2010 til 30-Dec-2014 (5-minut)	Data repræsentation + deep NN	NMSE, RMSE, MAE, MI

Kilde: Chong et. al. (2017).

[4]Tabellen er taget direkte fra Chong et al. (2017). Kildehenvisninger til de fremviste artikler kan findes i litteraturlisten.

Dette er ikke alene et karaktertræk ved deep learning vedrørende aktiemarkedet, tværtimod er denne eksperimentelle tilgang til modeltyper og struktur generel inden for stort set hele deep learning litteraturen.
Resultatet af dette projekts modellering kan da alene ikke anvendes som en vurdering af deep learning netværks anvendelighed til denne problemstilling. Derimod kan det konkluderes, at de netværksarkitekturer forsøgt anvendt i dette projekt ikke har kunne lære tilstrækkeligt af det givne inputsdata. Dette kan skyldes modelleringen i sig selv, men kan ligeså godt skyldes, mangel på signal i inputsdataet. Kunstige neurale netværk kan ikke lære noget fra data, som ikke har noget at lære af.

5.2.5 Reproducerbarhed og fortolkning af kunstige neurale netværk

En hyppigt kritiseret egenskab af neurale netværk er, at det ofte er udfordrende at rekreere det samme netværk. Dette skyldes, at der er en vis form for tilfældighed involveret[5] ved træningen af et neuralt netværk, da de tusindevis af parametre, som netværket justerer på under træningen, kan variere meget fra træning til træning. To netværk med præcis samme arkitektur (antal af skjulte lag, neuroner osv.) kan opnå vidt forskellige resultater under træningen, hvilket er kendt som *churn* (Cormier et al., 2016). De tilfældige forhold og initialværdier under træningen af et neuralt netværk er ikke en fejl men en bevidst og essentiel egenskab, der bidrager til, at neurale netværk lærer (Bhojanapalli et al., 2021).
Deep learning netværk bliver ofte sammenlignet med en sort boks, som er næsten umulig at få indsigt i. Dette skyldes, at det er ekstremt udfordrende at tolke på parametrene og vægtene, som et neuralt netværk sætter. Komplekse neurale netværk har ofte ti-tusindevis af parametre, hvis ikke flere[6]. Ligeledes er interaktionen mellem disse, og kompleksiteten, ikke klar, f.eks. om der er ikke-lineære forhold imellem variablene.
Dybe og komplekse neurale netværk har haft stor succes indenfor mange områder, men for det menneskelige øje vil det ses som en umulig opgave at tolke på sådanne modeller. For en investor opstår der et problem, hvis man ikke er i stand til at tolke på en afkastmodel, da det heller ikke vil være muligt at vurdere risikoen bag forecastene. Som diskuteret i afsnit 5.2.1, udvikler forholdende på de finansielle markeder sig konstant. En model, der historisk set har haft gode resultater, vil nødvendigvis ikke fortsætte med dette, hvis markedet ændrer sig. Det er da essentielt for en investor at kunne forstå, hvad en forecastingsmodel vægter

[5] F.eks. model initialisering, hvilket refererer til de vægte, netværket starter træningen ud fra. Se Bhojanapalli et al. (2021) for en kort gennemgang af de kilder, der bidrager til, at to identiske netværkstrukturer forudsiger forskellige resultater.

[6] F.eks. har netværk til billedgenkendelse ofte mange millioner parametre.

og ligeledes, hvad den ikke vægter højt til at forstå, om hvad end nye forhold, som måtte blive gældende på markedet, også vil blive fanget af modellen i dens forudsigelser.

6. Konklusion

I dette projekt er der blevet opstillet (i alt 30) RNN-modeller (Recurrent Neural Network), der alle anvender en LSTM (Long Short-Term Memory)-arkitektur med det formål at forudsige det fremtidige ugentlige aktieafkast under markedsforholdene i den globale Covid-19-pandemi. Analysen har foregået på ti forskellige aktier fra Dow Jones aktieindekset, hvor tre LSTM-netværk blev opstillet til forecasting af hver aktie. To af modellerne er trænet på sekvenser af 5-minuts data for henholdsvis 3 og 5 handelsdage. Det sidste kunstige neurale netværk er trænet på daglig kursdata i sekvenser af 90 handelsdage. Interessen var at undersøge hvorvidt LSTM-neurale netværk er i stand til at modellere aktiekurser under sådan en særligt volatil tidsperiode for aktiemarkedet. Forecastene på de ti aktier fra de tre kunstige neurale netværk blev anvendt i et MVO-framework til sammensætning af investeringsporteføljer i en out-of-sample test. Til dette formål blev variansen og kovariansen mellem de ti aktiver estimerede via en DCC-GARCH for at tage højde for tidsvarierende volatilitet. De sammensatte porteføljer blev da økonomisk evaluerede via en forvaltningsafgift sammenlignet med et baseline forecast fra en naiv random walk proces. Vores resultater viser, at porteføljerne sammensat baseret på forecast af de tre LSTM-modeller ikke formår at overgå porteføljeafkastet opnået med afkastforventinger fra den naive model. Dette er et resultat af, at de trænede kunstige neurale netværk ikke var i stand til akkurat at forecaste de volatile aktiekurser i perioden.
Resultaterne præsenteret i dette projekt kan i sig selv ikke anvendes til at demonstrere, hvorvidt deep learning netværk er egnede til at forecaste aktieafkast. Resultaterne skal ses i kontekst af den meget volatile periode samt opstillingen af netværkene og inputsdataet, de trænes på. Udfordringenerne kunstige neurale netværk står overfor, når det gælder modellering af kapitalmarkederne, lægger i stedet op til diskussionen om at udforske potentielle udvidelser af inputsdataet. Inputsdataet anvendt til træning af de kunstige neurale netværk har i dette projekt været forholdsvis simpelt, hvor udelukkende tidssekvenser af historisk kursdata er blevet anvendt. Et logisk næste skridt for denne modeltype vil være at introducere flere forklarende variable baserede på økonomisk teori til at gå i hånd med kursdataet. Grundet kompleksiteten af relationen mellem aktiepriser og faktorer, som økonomisk teori påpeger har en indflydelse på den fremtidige kursudvikling, er det i teorien oplagt at anvende deep learning teknikker til at modellere.

7. Litteraturliste

Abraham, A. (2005) *Artificial Neural Networks*, i bogen *Handbook of measuring system design* (2005) pp. 900-908, John Wiley Sons Ltd.

Arditti, F. D. (1967) *Risk and the Required Return on Equity*, The Journal of Finance, Vol. 22, No. 1, pp. 19-36

Arditti, F. D. & Levy, H. (1975) *Portfolio Efficiency Analysis in Three Moments: The Multiperiod Case*, The Journal of Finance, Vol. 30, No. 3, pp. 797-809

Arévalo, A., Niño, J., Hernández, G. & Sandoval, J. (2016) *High-Frequency Trading Strategy Based on Deep Neural Networks*, International Conference on Intelligent Computing, ICIC 2016: Intelligent Computing Methodologies, pp. 424-436

Arrow, K. J. (1965) *Aspects of the Theory of Risk Bearing*, Helsinki: Yrjo Jahnssonin Saatio.

Athayde, G. M. & Flores, R. G (2004) *Finding a maximum skewness portfolio - a general solution to three moments portfolio choice*, Journal of Economic Dynamics and Control, Vol. 28, Issue 7, pp. 1335-1352

Atsalakis, G. S. & Valavanis, K. P. (2013) *Surveying Stock Market Forecasting Techniques - Part I: Conventional Methods*, Computation Optimization in Economics and Finance Research Compendium, Nova Science Publishers Inc, pp. 49-104

Atsalakis, G. S. & Valavanis, K. P. (2009) *Surveying stock market forecasting techniques - Part II: Soft computing methods*, Expert Systems with Applications, Vol. 36, Issue 3, Part 2, pp. 5932-5941

Baviera, R. & Bianchi, G. (2019) *Model risk in mean-variance portfolio selection: an analytic solution to the worst-case approach*, Cornell University, arXiv:1902.06623v2

Beardsley, X. W., Field, B. & Xiao, M. (2012) *Mean-Variance-Skewness-Kurtosis Portfolio Optimization with Return and Liquidity*, Communications in Mathematical Finance, Vol. 1, No. 1, pp. 13-49

Bhojanapalli, S., Wilber, K., Veit, A., Rawat, A. S., Kim, S., Menon, A. & Kumar, S. (2021) *On the Reporducibility of Neural Network Predictions*, Google Research, New York

Bollerslev, T. (1986) *Generalized autoregressive conditional heteroskedasticity.* Journal of econometrics, Vol.31 (3), p.307-327

Bollerslev, T. (1990) *Modelling the coherence in short-run nominal exchange rates: A multivariate generalized ARCH model*, The Review of Economics and Statistics, vol. 72, issue 3 (Aug., 1990), pp. 498-505

Borch, K. (1969) *A Note on Uncertainty and Indifference Curves*, The Review of Economic Studies, Vol. 36, No. 1, pp. 1-4

Brandt, M. W. (2007) *Portfolio Choice Problems*, i Ait-Sahalia, Y. & Hansen, L. P. *Handbook of Financial Econometrics* (2009), North Holland

Briec, W., Kerstens, K. & Jokung, O. (2007) *Mean-Variance-Skewness Portfolio Performance Gauging: A General Shortage Function and Dual Approach*, Management Science, Vol. 53, No. 1, pp. 135-149

Cervelló-Royo, R., Guijarro, F. & Michniuk, K. (2015) *Stock market trading rule based on pattern recognition and technical analysis: Forecasting the DJIA index with intraday data*, Expert Systems with Applications, Vol. 42, Issue 14, pp. 5963-5975

Chen, T.-l. & Chen, F.-y. (2016) *An intelligent pattern recognition model for supporting investment decisions in stock market*, Information Sciences, Vol. 346, pp. 261-274

Chiang, W.-C., Enke, D., Wu, T. & Wang, R. (2016) *An adaptive stock index trading decision support system*, Expert Systems with Applications, Vol. 59, pp. 195-207

Chong, E., Han, C. & Park, F. C. (2017) *Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies*, Expert Systems with Applications, Vol. 83, pp. 187-205

Chourmouziadis, K. & Chatzoglou, P. D. (2016) *An intelligent short term stock trading fuzzy system for assisting investors in portfolio management*, Expert Systems with Applications, Vol. 43, pp. 298-311

Cohn, R. A., Lewellen, W. G, Lease, R. C. & Schlarbaum, G. G. (1975) *Individual Investor Risk Aversion and Investment Portfolio Composition*, The Journal of Finance, Vol. 30, No. 2, pp. 605-620

Cormier, Q., Fard, M. M., Canini, K. & Gupta, M. (2016) *Launch and iterate: REducing prediction churn*, Advances in Neural Information Processing systems 29, pp. 3179-3187

Corte, P. D., Sarno, L. & Tsiakas, I. (2008) *An Economic Evaluation of EMpirical Exchange Rate Models*, The Review of Financial Studies, Vol. 22, Issue 9, pp. 3491-3530

Dargan, S., Kumar, M., Ayyagari, M. R. & Kumar, G. (2019) *A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning*, Archives of Computational Methods in Engineering, Vol. 27, pp. 1071-1092

Enders, W. (2014) *Applied Econometric Time Series*, 4. udgave, John Wiley & Sons Inc

Engle, R. (2002) *Dynamic Conditional Correlation - A Simple Class of Multivariate GARCH Models*. Journal of Business Economic Statistics; Alexandria Vol. 20, Iss. 3: 339-350.

Enke, D. & Mehdiyev, N. (2013) *Stock Market Prediction Using a Combination of Stepwise Regression Analysis, Differential Evolution-based Fuzzy Clustering, and a Fuzzy Inference Neural Network*, Intelligent Automation & Soft Computing, Vol. 19, No. 4, pp. 636-648

Fama, E. F. (1965) *Random Walks in Stock Market Prices*, Financial Analysts Journal, Vol. 21, No. 5, pp. 55-59

Fama, E. F., Fisher, L. & Jensen, M. C. (1969) *The Adjustment of Stock Prices to New Information*, International Economic Review, Vol. 10, No. 1, pp. 1-21

Fama, E. F. (1970) *Efficient Capital Markets: A Review of Theory and Empirical Work*, the Journal of Finance, Vol. 25, No. 2, pp. 383-417

Fama, E. F. & French, K. R. (1993) *Common risk factors in the returns on stocks and bonds*, Journal of Financial Economics, Vol. 33, Issue 1, pp. 3-56

Feldstein, M. S. (1969) *Mean-Variance Analysis in the Theory of Liquidity Preference and Portfolio Selection*, The Review of Economic Studies, Vol. 36, No. 1, pp. 5-12

Fielitz, B. D. (1976) *Further Results on Asymmetric Stable Distributions of Stock Price Changes*, The Journal of Financial and Quantitative Analysis, Vol. 11, No. 1, pp. 39-55

Fleming, J., Kirby, C. & Ostdiek, B. (2001) *The Economic Value of Volatility Timing*, The Journal of Finance, Vol. 56. No. 1

Freitas, F. D., De Souza, A. F. & de Almeida, A. R. (2009) *Prediction-based portfolio optimization model using neural networks*, Neurocomputing, Vol. 72, Issue 10-12, pp. 2155-2170

Gers, F., Schmidhuber, J. & Cummins, F. (2000) *rLearning to Forget: Continual Prediction with LSTM*. Neural Comput 2000; 12 (10): 2451–2471

Ghalanos, A. (2019) *rmgarch: Multivariate GARCH models*. R package version 1.3-7.

Goodfellow, I., Bengio, Y. & Courville, A. (2016) *Deep Learning*. The MIT Press.

Green, J., Hand, J. R. M. & Zhang, X. F. (2013) *The Supraview of Return Predictive Signals*, Review of Accounting Studies 18, pp. 692-730

Gu, S., Kelly, B. T. & Xiu, D (2019) *Empirical Asset Pricing via Machine Learning*, Chicago Booth Research Paper No. 18-04, 31st Australasian Finance and Banking Conference 2018, Yale ICF Working Paper No. 2018-09

Guiso, L & Paiella, M. (2008) *Risk Aversion, Wealth, and Background Risk*, Journal of the European Economic Association, Vol. 6, No. 6, pp. 1109-1150

Harvey, C. R., Liu, Y. & Zhu, H. (2016) *...and the Cross-Section of Expected Returns*, the Review of Financial Studies, Vol. 29, Issue 1, pp. 5-68

Heaton, J. B., Polson, N. G & Witte, J. H. (2016) *Deep Learning in Finance*, Cornell University, arXiv:1602.06561v3

Hillier, D., Grinblatt, M. & Titman, S. (2011) *Financial Markets and Corporate Strategy*, McGraw-Hill, Second European Edition

Hinton, G. E. & Salakhutdinov, R. R. (2006) *Reducing the Dimensionality of Data with Neural Networks*, Science, Vol 313, Issue 5786, pp. 504-507

Kane, A. (1982) *Skewness Preference and Portfolio Choice*, The Journal of Financial and Quantitative Analysis, Vol 17, No. 1, pp. 15-25

Kelly, B., Israel, R. & Moskowitz, T. (2020) *Can Machines "Learn" Finance?*, Journal of Investment Management, Vol. 18, No. 2

Lai, T. (1991) *Portfolio selection with skewness: A multiple-objective approach*, Review of Quantitative Finance and Accounting 1(3), pp. 293-305

Li, S., Li, W., Cook, C., Zhu, C. & Gao, Y. (2018) *Independently Recurrent Neural Netowrk (IndRNN): Building A Longer and Deeper RNN*, Cornell University, arXiv:1803.04831v3

Markowitz, H. (1952) *Portfolio Selection*, The Journal of Finance, Vol. 7, No. 1 (Mar., 1952), pp. 77-91

Merton, R. C (1972) *An Analytic Derivation of the Efficient Portfolio Frontier*, Journal of Financial and Quantitative Analysis, Vol. 7, No. 4, pp. 1851-1872

Niaki, S. T. A., & Hoseinzade, S. (2013) *Forecasting S&P index using artificial neural networks and design of experiments*, Journal of Industrial Engineering International, Vol 9., No. 1

Olah, C. (2015) *Understanding LSTM Networks*, fundet på http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Orskaug, E. (2009) *Multivariate DCC-GARCH Model - With Various Error Distributions*, Norwegian University of Science and Technology

Patel, J., Shah, S., Thakkar, P. & Kotecha, K. (2015) *Predicting stock market index using fusion of machine learning techniques*, Expert Systems with Applications, Vol. 42, Issue 4, pp. 2162-2172

Pesaran, M. H. & Timmermann, A. (1992) *A Simple Nonparametric Test of Predictive Performance*, Journal of Business & Economic Statistics, Vol. 10, No. 4, pp. 461-465

Pratt. J. W. (1964) *Risk Aversion in the Small and in the Large*, Econometrica, Vol. 32, . 1/2, pp. 122-136

Qiu, M., Song, Y. & Akagi, F. (2016) *Application of artificial neural network for the prediction of stock market returns: The case of the Japanese stock market*, Chaos, Solitons & Fractals, Vol. 85, pp. 1-7

Ross, S. A. (1976) *The Arbitrage Theory of Capital Asset Pricing*, Journal of Economic Theory, Vol. 13, Issue 3, pp. 341-360

Samuelson, P. A. (1967) *General Proof that Diversification Pays*, The Journal of Financial and Quantitative Analysis, Vol. 2, No. 1, pp. 1-13

Scott, R. C. & Horvath, P. A. (1980) *On the Direction of Preference for Moments of Higher Order than the Variance*, The Journal of Finance, Vol 35., No. 4, pp. 915-919

Simkowitz, M. A. & Beedles, W. L. (1978) *Diversification in a Three-Moment World*, The Journal of Financial and Quantitative Analysis, Vol. 13, No. 5, pp. 927-941

Singleton, J. C. & Wingender, J. (1986) *Skewness Persistence in Common stock* The Journal of Financial and Quantitative Analysis, Vol. 21, No. 3, pp. 335-341

Zhong, X. & Enke, D. (2017) *Forecasting daily stock market return using dimensionality reduction*, Expert Systems with Applications, Vol. 67, pp. 126-139